253885-4-F

Final Report

UTILITY ANALYSIS OF HIGH-RESOLUTION MULTISPECTRAL IMAGERY

Volume 4: Image Based Sensor Model (IBSM) Version 2.0 Technical Description

M.T. Eismann, S.D. Ingle, and M. Slyz

Environmental Research Institute of Michigan
P.O. Box 134001
Ann Arbor, MI 48113-4001

March 1996

Sponsored by:

ASC/REFQ
2640 Loop Road West
Wright-Patterson AFB, OH 45433-7106

Contract No.: DLA900-88-D-0392
Delivery Order: 57

ERIM
P.O. Box 134001
Ann Arbor, MI 48113-4001

ERIM-320

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1996	

4. TITLE AND SUBTITLE
Utility Analysis of High-Resolution Multispectral Imagery, Volume 4: Image Based Sensor Model (IBSM) Version 2.0 Technical Description

5. FUNDING NUMBERS

6. AUTHOR(S)
M.T. Eismann, S.D. Ingle, and M. Slyz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Environmental Research Institute of Michigan
P.O. Box 134001
Ann Arbor, MI 48113-4001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ASC/REFQ
2640 Loop Road West
Wright-Patterson AFB, OH 45433-7106

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

The Image Based Sensor Model (IBSM) is a modular set of numerical tools for designing, evaluating, and modeling electro-optical and infrared (EO/IR) imaging sensors. The primary motivation which led to the development of IBSM was the need for a model which (a) could produce simulated sensor imagery (based on high fidelity input imagery) in addition to sensor performance metrics to better characterize the imaging performance of a sensor system, and (b) provides the flexibility to evaluate and compare sensors and imaging configurations with differing characteristics without rewriting computer code. The model operates within the Khoros Cantata environment, and can perform realistic simulation of image degradations and parametric modeling with a wide range of atmospheric, sensor, data link, and processing effects. This report provides a comprehensive overview of the model.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Performance Modeling, E/O Sensor, Infrared	135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-92)
Prescribed by ANSI Std. 239-18
298-102

PREFACE

This report documents a portion of the results of a study called "Utility Analysis of High-Resolution Multispectral Imagery" performed by the Electro-Optical Science Laboratory of the Environmental Research Institute of Michigan, Ann Arbor, Michigan, for the Air Force ASC/REFQ during the period January 1994 through May 1995. This study was performed under Delivery Order 57 within the Infrared Information Analysis Center (IRIA) program, contract number DLA900-88-D-0392, for which the Defense Electronic Supply Center (DESC), Dayton, Ohio, serves as the contracting agency. The ASC program manager was Doug Amlin. The ERIM program manager was Michael T. Eismann. The authors of this report are Michael T. Eismann, Stephen D. Ingle, and M. Slyz.

CONTENTS

FIGURES

TABLES

1.0 INTRODUCTION

The Image Based Sensor Model (IBSM) is a modular set of numerical tools for designing, evaluating, and modeling electro-optical and infrared (EO/IR) imaging sensors. The primary motivation which led to the development of IBSM was the need for a model which (a) could produce simulated sensor imagery in addition to sensor performance metrics to better characterize the imaging performance of a sensor system, and (b) provides the flexibility to evaluate and compare sensors and imaging configurations with differing characteristics without rewriting computer code.

To satisfy the first need, IBSM is primarily geared to image simulation; that is, it can generate renditions of high fidelity input scenes which are representative of actual sensor imagery, including a wide range of atmospheric, sensor, data link, and processing effects. In addition to this image-in, image-out mode of operation, the model also contains tools for parametrically computing sensor performance metrics such as signal-to-noise ratio (SNR), ground sample distance (GSD), ground resolved distance (GRD), and image quality as defined by the National Image Interpretability Rating Scale (NIIRS). The combination of these two operating modes provides the user the means for evaluating the level of synergy between subjective imaging performance to quantitative sensor performance metrics.

To satisfy the second need, IBSM was designed to be completely modular in that each effect modeled is done so as a self-contained function. This provides the user with a set of modeling building blocks which can be tied together in a wide variety of ways to easily model almost any sensor design or imaging configuration. Each module contains its own graphical user interface (GUI) for simple entry of the defining module parameters.

The modular design philosophy also has the advantage that it easily accommodates upgrades to extend it to sensing and processing effects not currently covered. This simply involves the generation of a new building block as opposed to a full model revision. As an example, the current model does not contain tools for synthetic scene generation, but such a capability could be added without impacting the remainder of the model.

This report overviews the IBSM model, including a brief overview (Section 2), detailed description (Section 3), usage overview (Section 4), example models (Section 5), and software overview (Section 6).

2.0 MODEL OVERVIEW

2.1 MODELING APPROACH

IBSM consists of an extensive set of discrete modeling tools which can be tied together to evaluate the performance of a wide variety of sensors. IBSM was primarily intended to be used in an image simulation mode as depicted in Figure 2-1. Such a simulation begins with a high fidelity input image representing the target and/or background scene to be simulated. The image is then degraded through a series of operations characterizing the intervening atmosphere, the sensor optics, the electro-optical detector, the data link, and subsequent signal processing. Various utilities for image format conversion, image/spatial spectrum conversion, metric computation, and image display also exist.

A parametric modeling capability is also supported by IBSM, specifically for computing image quality. This is depicted in Figure 2-2. As opposed to the image simulation mode, parametric modeling does not explicitly involve an image chain. Rather, sensor metrics such as the modulation transfer function (MTF), signal-to-noise ratio (SNR), ground resolved distance (GRD), and image quality (NIIRS) are computed directly from sensor parameters. This computation can be performed iteratively while altering a specified parameter within the loop to investigate performance as a function of an arbitrary independent variable.

In general, the model incorporates multispectral imagery (MSI). In the image simulation mode, MSI are explicitly carried through the image chain. Additionally, the center wavelengths and spectral bandwidths of each band are carried through the chain such that wavelength dependent modules operate differently on the image bands. Within each band, however, a narrowband assumption is usually made. That is, the band center wavelength defines the spectral dependence. The parametric modeling tools, however, explicitly integrate over the sensor bandwidth.

Most modules are also based on a linear, space invariant assumption. For example, many effects are characterized by an optical transfer function (OTF) which operates on the spatial Fourier transform of an image. Nonlinear and space variant effects, however, are not necessarily incompatible with the basic model formulation.

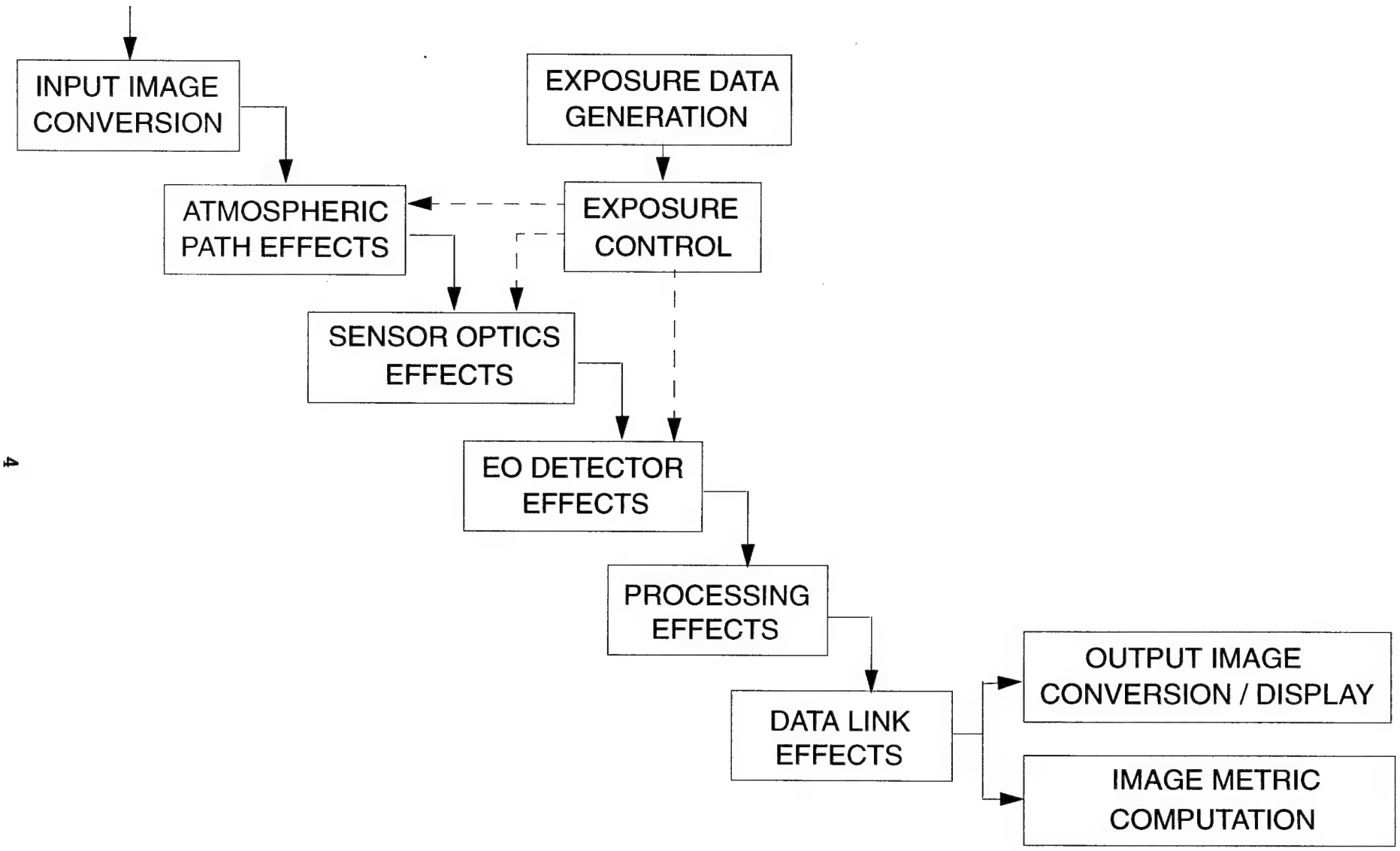

Figure 2-1: Image Simulation Mode

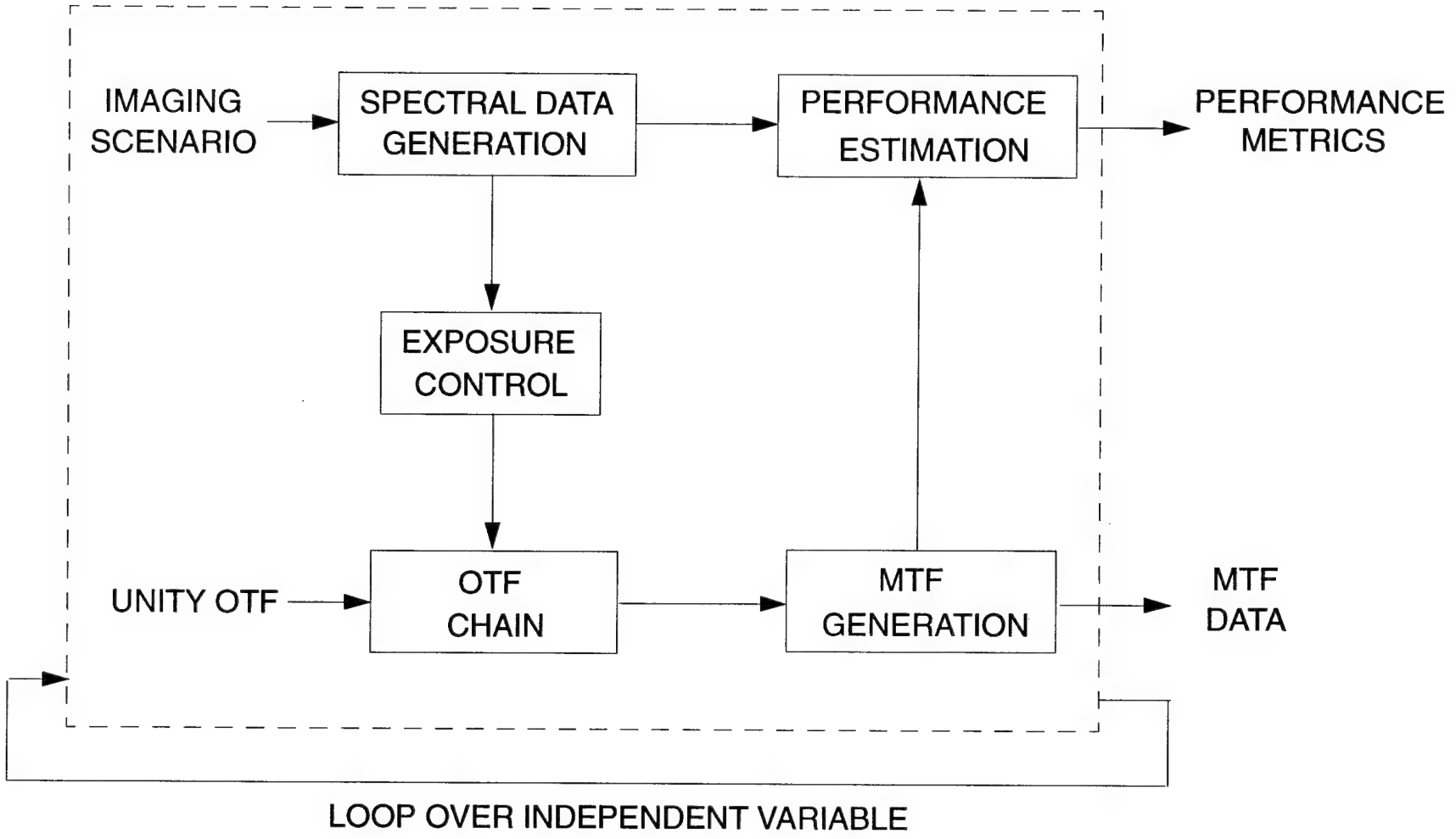

Figure 2-2: Parametric Performance Mode

Table 2-1: IBSM Modules

ATMOSPHERIC EFFECTS

MODTRAN Input
Path Trans/Rad
Turbulence OTF
Aero-Optic OTF

SENSOR OPTICS EFFECTS

Diffraction OTF
Defocus OTF
Jitter OTF
Drift OTF
Wavefront OTF
User OTF (1-D)
User OTF (2-D)
Radiometry

E/O DETECTOR EFFECTS

Detector Size OTF
TDI OTF
CTE OTF
Diffusion OTF
Detection
Noise
Quantization
Sampling
Nonuniformity

PROCESSING EFFECTS

Image Interlace
Aggregation

DATA LINK EFFECTS

JPEG Compress
JPEG Decompress
TSVQ Compress/Decompress
Wavelet Comp/Decomp
Finite Buffer
Channel Errors

PARAMETRIC PERFORMANCE

Create Spectra
Create MTF
Sensor Performance
Exposure Control
Image Quality

UTILITIES

VIFF to IBSM
List IBSM File Data
Transform
Create Plot Data
Create Radiance Image
Ground/Angle Transform
Load Variables
XGraph Plot

2.2 MODELING ENVIRONMENT

IBSM has been upgraded from an earlier version [1] to operate as a set of toolboxes within the Khoros 2.x Cantata environment. Cantata is an image and signal processing prototyping environment which is widely used throughout the government and university research communities. What Cantata provides is the flexible modeling environment and graphical user interface (GUI) which makes IBSM relatively easy to use. In addition, Cantata provides a large set of standard image and signal processing and display utilities which are at the user's disposal.

The IBSM software consists of several functional layers of code written in the C programming language. The executive layer was written using the Khoros development system, and includes all the Khoros-specific functions, including file I/O and graphical user interface. The Khoros library function layer consists of the interface between the Khoros executive code and the sensor modeling library, which contains the fundamental modeling routines. The sensor modeling library is a set of C subroutines which has been written independent of Khoros or any other executive code. Numerical Recipes and MODTRAN are used for analytical computations and atmospheric modeling.

2.3 CAPABILITIES SUMMARY

IBSM currently consists of six toolboxes: *Atmosphere*, *Optics*, *Detector*, *Data Link*, *Performance*, and *Utilities*. Each of these toolboxes are briefly described here. A more detailed description is given in Section 3.

The *Atmosphere* toolbox contains the tools for simulating image propagation along the atmospheric path between the imaged area and the sensor, including atmospheric path transmission and radiance, long path turbulence, and aero-optical boundary layer turbulence. Atmospheric path transmission and radiance effects are computed by executing MODTRAN. A separate module was written to provide a graphical user interface to MODTRAN and allow simplified entry of all the defining atmospheric parameters.

The *Optics* toolbox contains the tools for characterizing the image effects of an optical sensor system. These effects are primarily modeled as optical transfer functions (OTFs) and, therefore, generally operate on a spatial spectrum (Fourier transform of an image) rather than the image itself. OTFs are included for aperture diffraction, defocus,

wavefront irregularity, line-of-sight drift, and line-of-sight jitter. Modules for incorporating user defined OTF data in the form of one-dimensional and two-dimensional OTFs or point spread functions (PSF) are also available. Finally, a radiometric tool is provided for transforming a pupil plane radiance image into a focal plane irradiance image, including stray radiance sources.

The *Detector* toolbox contains the tools for characterizing the electro-optical detection process at the focal plane. Once again, several of the effects are modeled as OTFs, including that due to the finite detector element size, the time-delay-integrate (TDI) process, charge transfer inefficiency, and carrier diffusion. Conversion of the focal plane irradiance image to a sampled electronic image is performed by the combination of a photoelectronic detection and sampling tool. The non-ideal performance of the detector can be characterized by noise, quantization, and nonuniformity modules.

The *Performance* toolbox contains the modules written primarily for parametric sensor modeling. This includes modules for generating atmospheric and sensor spectral profiles; computing MTF from an OTF chain; computing SNR, GRD, and/or NIIRS directly from sensor parameters; computing line response functions, SNR, and NIIRS from an image; and configuring a sensor for proper exposure control.

The *Processing* toolbox contains modules not provided by the standard Khoros library for image processing. This specifically includes functions for combining image interlaces and aggregating pixels and/or bands.

The *Data Link* toolbox contains the tools for introducing the effects of lossy data compression, data underflows and overflows due to finite transmission buffers, and channel bit errors associated with transmitting sensor image data across a real communication link.

The *Utilities* toolbox contains routines for image processing, file format conversion, and also for transforming images into spatial spectra, and vice versa.

A complete list of IBSM modules is given as Table 2-1.

3.0 DETAILED DESCRIPTION

This section contains a detailed description of the physical basis of the IBSM toolboxes. Before proceeding, however, some common nomenclature will be defined. For routines which operate on radiance imagery, $L_{in}(x,y)$ and $L_{out}(x,y)$ will represent the image radiance as a function of spatial coordinates x and y. Analogous notation will be used for reflectance imagery $\rho(x,y)$, temperature imagery $T(x,y)$, irradiance imagery $E(x,y)$, and imagery in detected photoelectron units $N(x,y)$. The spatial coordinates are generally specified in angular (radian) units unless otherwise noted.

Optical transfer functions are characterized by $H(u,v)$ and operate on spatial spectra $G_{in}(u,v)$ to form $G_{out}(u,v)$ such that

$$G_{out}(u,v) = H(u,v)\, G_{in}(u,v) \tag{3-1}$$

where u,v are spatial frequency (rad^{-1}) dimensions. In general, the OTF and the spatial spectra are complex (real and imaginary) two-dimensional distributions. The units of $G(u,v)$ are generally arbitrary and $H(u,v)$ is dimensionless.

In the case of multiband images or spatial spectra, the basic functions outlined in this section operate separately on each band. This is implied in the descriptions to follow, particularly where the operations are wavelength dependent. Each image band is characterized by its center wavelength λ_0 and spectral bandwidth $\Delta\lambda$.

In the Khoros implementation of IBSM, image data is passed between modules via files. Generally, the imagery is in floating-point format and scaled to certain physical units. In addition, ancillary information is contained in the files such as scaling information, spectral band centers and bandwidths, and data type information. This is included in the Khoros file structure as *attributes*. The need for this additional information in the input file is denoted for each of the modules described.

3.1 ATMOSPHERE

3.1.1 MODTRAN Input

This function provides a user interface for generating a MODTRAN path data file as an input to IBSM functions that execute MODTRAN, including *Path Trans/Rad*, *Create Spectra*, and *Create Radiance Image*. The output file is in ASCII format, as given in Table 3-1. All MODTRAN-related functions have been upgraded for compatibility with MODTRAN3.

The user is referred to the applicable MODTRAN documents [2, 3] for an overview of the model inputs. A few comments regarding commonly used defaults and overrides are given here. The visibility is determined from the haze model if a zero value is entered, otherwise it is overridden by the user entry. The surface albedo can be explicitly entered (assumed spectrally flat), or determined from one of the spectral profiles which exist. Finally, the target zenith angle is really measured from zenith (MODTRAN definition) such that -180° is a downlooking sensor.

All modules which are based on MODTRAN require a path data file and an optional working directory as an input. The module creates a temporary working directory (if none is specified); copies the DIRAC, sun1, sun2, and refbkg files to the working directory; forms the TAPE5 input file; executes MODTRAN3 (location defined by MODTRAN_HOME environment variable); and extracts the desired data from the generated TAPE6 or TAPE7 files.

3.1.2 Path Trans/Rad

This function applies the effects of atmospheric path transmission and radiance globally to an input radiance image. This is performed in two steps. First, the atmospheric transmission $\tau(\lambda)$ and path radiance $L_{path}(\lambda)$ as a function of wavelength are computed by MODTRAN based on the path data file created by the MODTRAN *Input* function. Then, the band-average transmission and path radiance is applied to the image

$$L_{out}(x,y) = \frac{L_{in}(x,y)}{\Delta\lambda} \int_{\Delta\lambda} \tau(\lambda)d\lambda + \int_{\Delta\lambda} L_{path}(\lambda)d\lambda \qquad (3\text{-}2)$$

Table 3-1: Format of MODTRAN Path Data File

MODEL	integer	Model atmosphere (1-6)
IMULT	integer	Multiple scattering (0 = off, 1 = on)
TBOUND	float	Surface temperature (K, 0.0 = first layer temp.)
SALB	float	Surface albedo (0-1, -1, -2, -3, -4, -5, -6)
IHAZE	integer	Aerosol haze model (0-6, 8-10)
ICLD	integer	Cloud model (0-10, 18-20)
VIS	float	Visibility (km, 0.0 = default)
RAINRT	float	Rain rate (mm/hr)
GNDALT	float	Ground altitude (km)
IDAY	integer	Day of the year (1-365)
ISOURC	integer	Celestial source (0 = sun, 1 = moon)
ANGLEM	float	Moon phase angle (deg)
H1	float	Target altitude (km)
H2	float	Sensor altitude (km)
ANGLE	float	Sensor zenith angle from target (deg)
PARM1	float	Sun azimuth angle from sensor LOS (deg)
PARM2	float	Sun zenith angle from target (deg)
IV1	integer	Minimum wavenumber (cm^{-1})
IV2	integer	Maximum wavenumber (cm^{-1})
IDV	integer	Wavenumber sampling (cm^{-1})
IFWHM	integer	Wavenumber resolution (cm^{-1})

It is imperative that the wavelength limits specified in the MODTRAN path data file cover each spectral band specified for the input image (attribute). Spatial scaling attributes for the input image are ignored. The input image must be in radiance (W/m^2sr) units.

3.1.3 Turbulence OTF

This function applies the effects of long path atmospheric turbulence to an image. It is characterized as an OTF and, therefore, operates in the spatial spectrum domain. The OTF is given by [4]

$$H(u,v) = e^{-3.44(\lambda_0 \rho / r_0)^{5/3}[1-\alpha(\lambda_0 \rho / D)^{1/3}]} \tag{3-3}$$

where D is the effective aperture diameter,

$$\rho = \sqrt{u^2 + v^2} \tag{3-4}$$

and r_0 is the correlation diameter defined by Fried [5, 6].

The correlation diameter can be either user specified or computed from an index structure parameter model along the line-of-sight path. In the latter case [7]

$$r_0 = 2.1\left[\frac{5.84\pi^2}{\lambda_0^{\,2}\cos\phi}\int_{h1}^{h2} C_n^{\,2}(h)\left(\frac{h-h_1}{h_2-h_1}\right)^{5/3} dh\right]^{-3/5} \tag{3-5}$$

where h_1 is the target altitude, h_2 is the sensor altitude, ϕ is the target nadir angle, and $C_n^2(h)$ is the vertical index structure parameter distribution.

Three $C_n^2(h)$ models are implemented, all of which are parametrized by the index structure parameter at h = 1 meter. This generally ranges from $10^{-12}\ m^{-2/3}$ representing fairly turbulent conditions to $10^{-14}\ m^{-2/3}$ representing fairly benign conditions [8]. The first model is a simple model [7]

$$C_n^{\,2}(h) = \begin{cases} C_n^{\,2}(h=1m) & h < 1m \\ \dfrac{C_n^{\,2}(h=1m)}{h} & h > 1m \end{cases} \tag{3-6}$$

The second is a model proposed by Hufnagel [9] for medium to high altitudes, and is extended to lower altitudes using a power coefficient to match the specified $C_n^2(h)$ at h = 1 m [7]. The functional form is

$$C_n^2(h) = \begin{cases} C_n^2(h=1m) & h<1m \\ C_n^2(h=1m)h^{-\left[\frac{15.85+\log C_n^2(h=1m)}{3}\right]} & 1m<h<1km \\ 8.2x10^{-56}v^2h^{10}e^{-h/1000}+2.7x10^{-16}e^{-h/1500} & h>1km \end{cases} \quad (3\text{-}7)$$

where v is the upper atmospheric wind speed (generally in the 18-36 m/s range). The final model is an approach based on empirical data [10]

$$C_n^2(h) = \begin{cases} C_n^2(h=1m) & h<1m \\ C_n^2(h=1m)h^{-1.7} & h<10m \\ 0.1C_n^2(h=1m)h^{-0.65} & h<1km \\ 3.16C_n^2(h=1m)h^{-1.15} & h<20km \\ 0 & h>20km \end{cases} \quad (3\text{-}8)$$

Figures 3-1 to 3-3 illustrate the index structure parameter models for a range of conditions: benign ($C_n^2 = 10^{-14}$ $m^{-2/3}$ at h = 1 m), moderate ($C_n^2 = 10^{-13}$ $m^{-2/3}$ at h = 1 m), and turbulent ($C_n^2 = 10^{-12}$ $m^{-2/3}$ at h = 1 m). For the Hufnagel model, upper atmosphere wind speeds of 18, 27, and 36 m/s were used, respectively.

The α parameter in Eq. (3-3) specifies whether the OTF is of the long exposure (α = 0) or short exposure (α = 0.5) variety. This is determined by the temporal nature of the turbulence relative to the sensor integration time. In general, the OTF is modeled as a mix of the long exposure H_L (u,v) and short exposure H_S (u,v) OTFs given by

$$H(u,v) = H_S(u,v)e^{-VT_d/r_0} + H_L(u,v)(1-e^{-VT_d/r_0}) \quad (3\text{-}9)$$

where T_d is the pixel dwell (or integration time) and V is the effective velocity of the atmosphere relative to the sensor pupil plane axes.

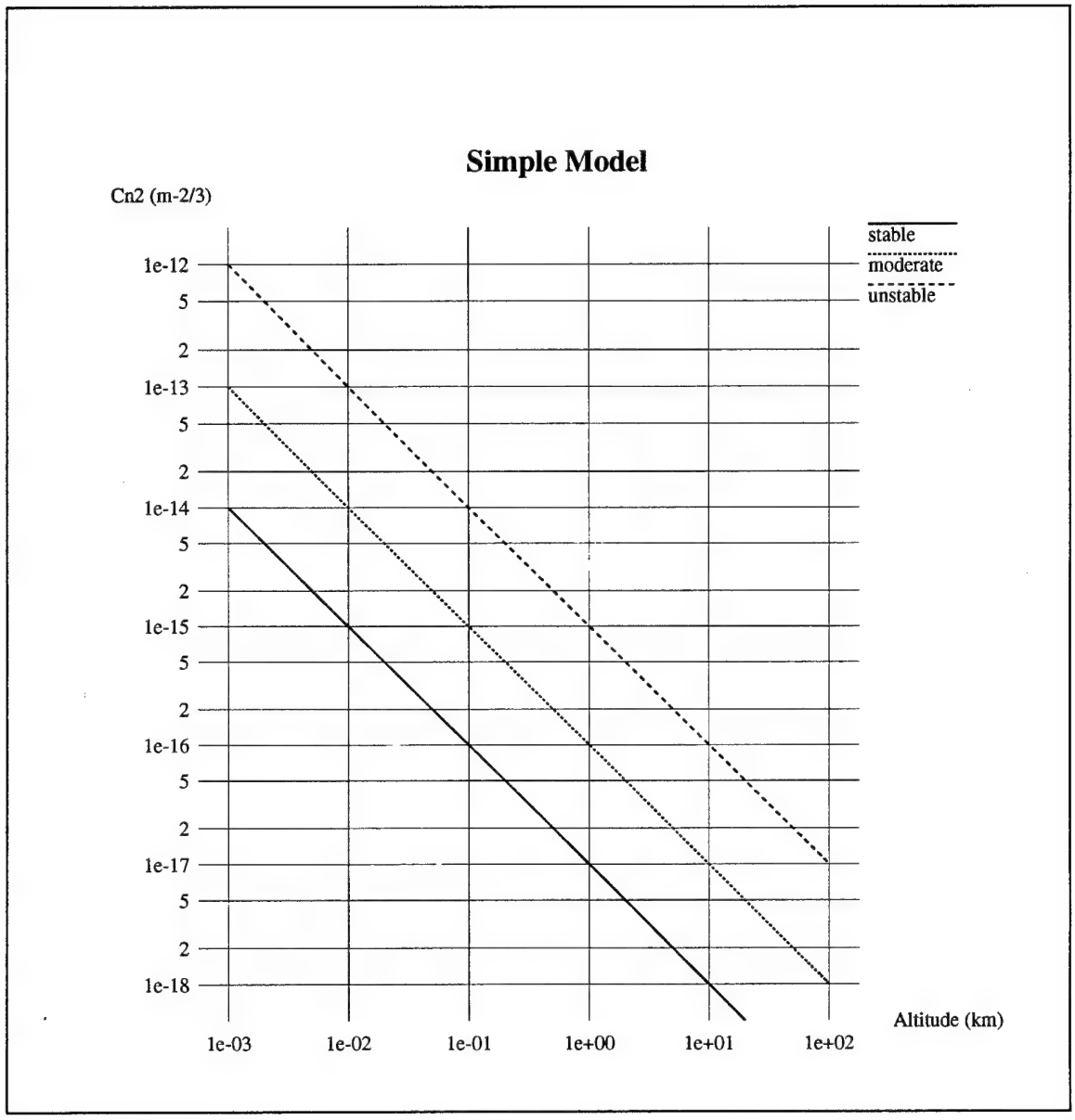

Figure 3-1: Simple Index Structure Parameter Model

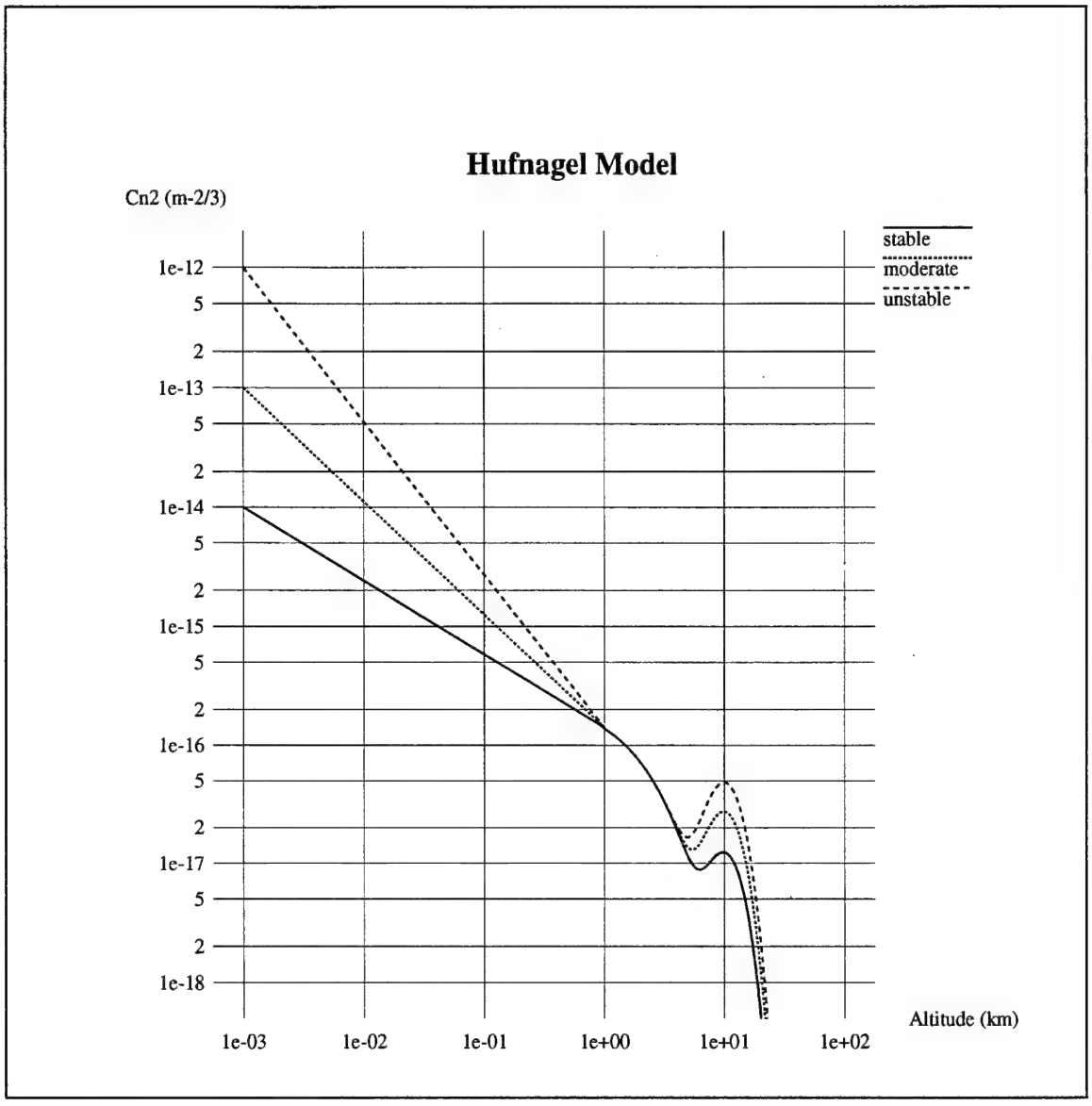

Figure 3-2: Hufnagel Index Structure Parameter Model

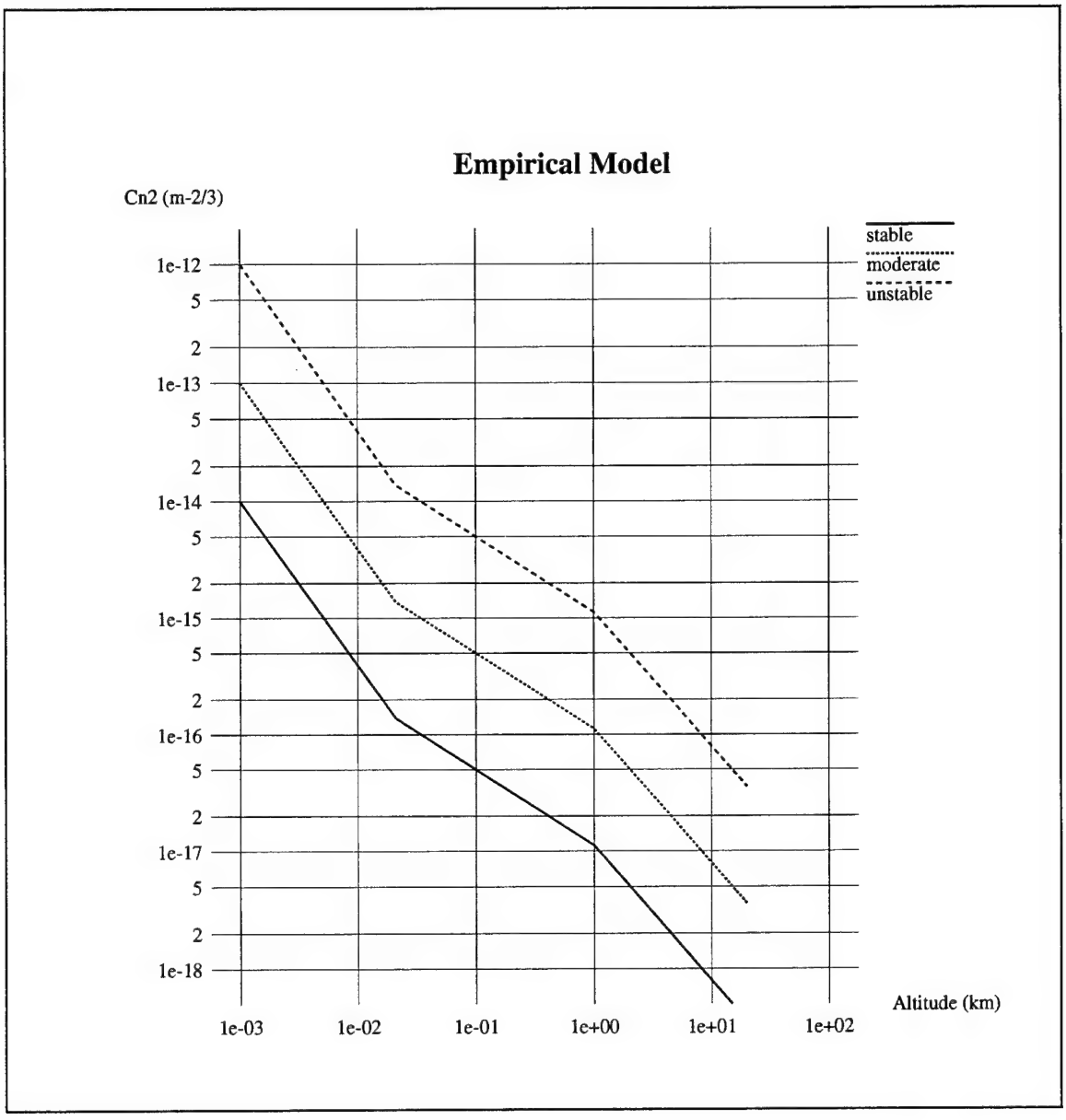

Figure 3-3: Empirical Index Structure Parameter Model

Spatial scaling (rad^{-1} units) and spectral band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.1.4 Aero-Optic OTF

This function applies the effects of boundary layer aero-optical turbulence to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The OTF is given by [11]

$$H(u,v) = e^{-\sigma_\phi^2\left[1-e^{-\lambda_0^2\left[\frac{u^2}{L_x^2}+\frac{v^2}{L_y^2}\right]}\right]} \tag{3-10}$$

where σ_ϕ^2 is the turbulence-induced wavefront phase variance and L_x, L_y are the spatial correlation lengths of the turbulent boundary layer.

The phase variance is given by

$$\sigma_\phi = \frac{2\pi}{\lambda_0} G\rho' \sqrt{2L_z t_{b1}} \tag{3-11}$$

where G is the Gladstone-Dale coefficient (0.22 x 10^{-3} m^3/kg), ρ' is the rms density fluctuation, L_z is the correlation length of the boundary layer in the normal direction, and t_{b1} is the boundary layer thickness.

The model is based on a "rule-of-thumb" equation for the rms density fluctuation and correlation lengths [12]. Specifically,

$$\rho' = 0.2\left[\rho_0 - \rho\right] \tag{3-12}$$

where,

$$\rho = \frac{\rho_0}{1 + r(\gamma - 1)M^2 / 2} \tag{3-13}$$

ρ_0 is the free stream air density, r = 0.89, γ = 1.4, and M is the Mach number (v/305 m/s). The correlation lengths are approximated by

$$L_x = 0.4\, t_{b1} \tag{3-14}$$

$$L_y = L_z = 0.2\, t_{b1} \tag{3-15}$$

The boundary layer is then specified by its thickness t_{b1} along with the free stream air density and platform velocity. Table 3-2 provides a typical free stream air density as a function of altitude.

Spatial scaling (rad^{-1} units) and spectral band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2 IBSM OPTICS

3.2.1 Diffraction OTF

This function applies the effects of aperture diffraction to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. For an unobscured circular aperture [13]

$$H(u,v) = \begin{cases} \dfrac{2}{\pi}\left[\cos^{-1}\left(\dfrac{\rho}{\rho_0}\right) - \dfrac{\rho}{\rho_0}\sqrt{1-\left(\dfrac{\rho}{\rho_0}\right)^2}\right] & \rho \le \rho_0 \\ 0 & \rho > \rho_0 \end{cases} \tag{3-16}$$

where ρ is defined by Eq. (3-4) and

$$\rho_0 = \frac{D}{\lambda_0} \tag{3-17}$$

where D is the effective aperture diameter. For a rectangular aperture [13]

Table 3-2: Typical Free Stream Air Density as a Function of Attitude

Altitude (km)	Air Density (kg/m^3)
1	1.0
2	0.9
5	0.7
10	0.4
12	0.3
15	0.2
20	0.1

$$H(u,v) = \Lambda\left(\frac{\lambda u}{d_x}\right)\Lambda\left(\frac{\lambda u}{d_y}\right) \tag{3-18}$$

where d_x, d_y are the aperture widths in their respective directions and

$$\Lambda(\xi) = \begin{cases} 1-|\xi| & |\xi| \le 1 \\ 0 & |\xi| > 1 \end{cases} \tag{3-19}$$

In the case of an obscuration, only the circular symmetric case is modeled (obscuration input is ignored for rectangular aperture). In this case, the OTF is given by [14]

$$H(u,v) = \frac{A+B+C}{1-\eta^2} \tag{3-20}$$

$$A = \begin{cases} \frac{2}{\pi}\left[\cos^{-1}\left(\frac{\rho}{\rho_0}\right) - \frac{\rho}{\rho_0}\sqrt{1-\left(\frac{\rho}{\rho_0}\right)^2}\right] & \rho \le \rho_0 \\ 0 & \rho > \rho_0 \end{cases} \tag{3-21}$$

$$B = \begin{cases} \frac{2\eta^2}{\pi}\left[\cos^{-1}\left(\frac{\rho}{\eta\rho_0}\right) - \frac{\rho}{\eta\rho_0}\sqrt{1-\left(\frac{\rho}{\eta\rho_0}\right)^2}\right] & \rho \le \eta\rho_0 \\ 0 & \rho > \eta\rho_0 \end{cases} \tag{3-22}$$

$$C = \begin{cases} -2\eta^2 & \rho < \frac{1-\eta}{2}\rho_0 \\ \frac{2\eta}{\pi} - \sin\phi + \left(\frac{1+\eta^2}{\pi}\right)\phi - \frac{2(1-\eta^2)}{\pi}\tan^{-1}\left[\left(\frac{1+\eta}{1-\eta}\right)\tan\left(\frac{\phi}{2}\right)\right] & \frac{1-\eta}{2}\rho_0 \le \rho \le \frac{1+\eta}{2}\rho_0 \\ 0 & \rho > \frac{1+\eta}{2}\rho_0 \end{cases} \tag{3-23}$$

$$\phi = \cos^{-1}\left[\frac{1+\eta^2-(2\rho/\rho_0)^2}{2\eta}\right] \tag{3-24}$$

ρ is defined in Eq. (3-4), $\rho_0 = D/\lambda_0$, and η is the relative obscuration (ratio of obscuration to aperture diameters).

Spatial scaling (rad^{-1} units) and spectral band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.2 Defocus OTF

This function applies the effects of optics defocus or blur to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The defocus (or blur) function is modeled as a gaussian distribution for which the 1/e blur radius in angular space is specified independently for the x and y axes. Mathematically [15]

$$H(u,v) = e^{\frac{-\pi^2}{4}(w_x^2 u^2 + w_y^2 v^2)} \tag{3-25}$$

where w_x and w_y are the 1/e blur spot radii in x and y.

Although not exact, this module can be used to model OTF degradations due to axial misfocus. The actual blur function will be dependent on the aperture characteristics, but a gaussian blur function is a fair approximation. For an axial defocus defined in the object domain (focus set to R_0, object at R)

$$w_{x,y} = 0.62D\left(\frac{1}{R} - \frac{1}{R_0}\right) \tag{3-26}$$

For an axial defocus Δz defined in the image domain,

$$w_{x,y} = \frac{0.62D\Delta z}{f^2} \tag{3-27}$$

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.3 Jitter OTF

This function applies the effects of wideband (with respect to the sensor integration time) jitter of the optical sensor line-of-sight to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. By virtue of the wideband assumption, the jitter is modeled as a zero mean, gaussian random process resulting in the gaussian expected value for the OTF [16]:

$$H(u,v) = e^{-2\pi\left(\sigma_x^2 u^2 + \sigma_y^2 v^2\right)} \tag{3-28}$$

where σ_x and σ_y are the rms jitter amplitudes in their corresponding angular directions.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.4 Drift OTF

This function applies the effects of constant angular drift of a sensor line-of-sight to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The OTF is defined by

$$H(u,v)=\text{sinc}(a_x u)\text{sinc}(a_y v) \tag{3-29}$$

where

$$\text{sinc}(\xi) = \frac{\sin(\pi\xi)}{\pi\xi} \tag{3-30}$$

and a_x and a_y are the angular drift magnitudes over the sensor integration time in their corresponding directions.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.5 Wavefront OTF

This function applies the effects of a spatially random wavefront error to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The OTF is given by [17]

$$H(u,v) = e^{-\sigma_\phi^2\left[1-e^{-\lambda^2\left[\frac{u^2}{L_x^2}+\frac{v^2}{L_y^2}\right]}\right]} \tag{3-31}$$

where the wavefront error is defined in the sensor pupil plane by the phase variance σ_ϕ^2 and x and y correlation lengths L_x and L_y (defined in spatial aperture dimensions). The phase variance is specified by an rms wavefront error (waves) at a reference wavelength and scaled to the image band center wavelength(s) in inverse proportion to the wavelength.

Spatial scaling (rad^{-1} units) and spectral band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.6 User OTF (1-D)

This function applies an OTF specified by one-dimensional user data to an image. As it is an OTF, it operates in the spatial spectrum domain. The data is contained in an ASCII file in formats accommodating real and complex data specified in radial form (radial), with separability in the x and y axes (x and y), or with separability and similarity in the x and y directions (x or y). The user data is interpolated using linear or spline techniques to the input spatial frequency grid prior to forming the 2-D OTF.

The various file formats characterizing the user OTF are given in Table 3-3. In the case of *radial* data, the OTF is interpolated onto the (u,v) grid at each point relative to its radius defined by Eq. (3-4). For *x and y* data, the input data is interpolated onto two functions $H_x(u)$ and $H_y(v)$, and the two-dimensional OTF is defined by

$$H(u,v) = H_x(u)H_y(v) \tag{3-32}$$

Table 3-3: User OTF (1-D) File Formats

SF(1)	$OTF_{real}(1)$	$[OTF_{imag}(1)]$
•		•
•		•
•		•
SF(N)	$OTF_{real}(N)$	$[OTF_{imag}(N)]$

(a) Radial Data

SF(1)	$OTF_{real}(1)$	$[OTF_{imag}(1)]$
•		•
•		•
•		•
SF(N)	$OTF_{real}(N)$	$[OTF_{imag}(N)]$

(b) X and Y Data

$SF_X(1)$	$OTF_{X,real}(1)$	$[OTF_{X,imag}(1)]$	$SF_Y(1)$	$OTF_{Y,real}(1)$	$[OTF_{Y,imag}(1)]$
•					•
•					•
•					•
$SF_X(N)$	$OTF_{X,real}(N)$	$[OTF_{X,imag}(N)]$	$SF_Y(N)$	$OTF_{Y,real}(N)$	$[OTF_{Y,imag}(N)]$

(c) X or Y Data

[] = data element exist only for complex data
N lines of space delimited data in all cases
SF = Spatial Frequency (rad^{-1})
OTF = Optical Transfer Function

For *x or y* data, $H_x(u)$ and $H_y(v)$ are identical by definition. Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.2.7 User OTF (2-D)

This function applies an OTF specified by two-dimensional user data to an image. As it is an OTF, it operates in the spatial spectrum domain. The user data containing scaling information is contained in an image file. This data can correspond to an OTF (rad^{-1} scaling attributes) or a point spread function (PSF, rad scaling attributes).

In the case of OTF data, the user defined data is interpolated onto the input spatial frequency grid using a bilinear technique. In the case of PSF data, the user OTF is computed at each grid point by

$$H(u,v) = \frac{1}{2\pi} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} h(i,j)\, e^{-i2\pi(x_i u + y_i v)} \tag{3-33}$$

where x_i and y_j are the angular positions corresponding to the PSF indices. If specified, the OTF data is normalized to unity at zero spatial frequency in either case.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced using the *Transform* tool.

3.2.8 Radiometry

This function performs a radiometric conversion of an effective pupil plane radiance image to a focal plane irradiance image, including the sensor throughput, thermal radiation of the optical chain, and stray radiance:

$$E(x,y) = \frac{\pi}{4(f/\#)^2}\left[\tau_{opt} L(x,y) + \varepsilon_{opt} L_{BB}\left(\lambda, T_{opt}\right)\Delta\lambda + L_s\right] \tag{3-34}$$

where (f/#) is the effective sensor F-number, τ is the optical train transmission including obscuration, ε_{opt} is the optical train effective emissivity, T_{opt} is the optical train effective

temperature, L_s is the in-band stray radiance (non-thermal), and L_{BB} is a blackbody spectral radiance function

$$L_{BB}(\lambda,T)=\frac{2hc^2}{\lambda^5}\frac{1}{e^{hc/\lambda KT}-1} \tag{3-35}$$

Spectral scaling attributes must be available in the input file header, and the image data must be in radiance (W/m^2sr) units. The output image data is in irradiance (W/m^2) units.

3.3 IBSM DETECTOR

3.3.1 Detector Size OTF

This function applies the effects of the finite detector width to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The detector is assumed to have rectangular form, which generates an OTF with a *sinc* distribution [18]. It is extended, however, to include pixel aggregation effects:

$$H(u,v)=\mathrm{sinc}\left(\frac{w_x u}{f}\right)\mathrm{sinc}\left(\frac{w_y v}{f}\right)\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}\cos\phi_{i,j} \tag{3-36}$$

where w_x and w_y are the detector sizes in x and y, f is the optical system focal length, N_x and N_y are the number of aggregated pixels in x and y,

$$\phi_{ij}=2\pi\left[\frac{ip_x u}{f}+\frac{jp_y v}{f}-\frac{(N_x-1)p_x u}{2f}-\frac{(N_y-1)p_y v}{2f}\right] \tag{3-37}$$

and p_x and p_y are the detector center-to-center spacings (pitch) in x and y. The summation in Eq. (3-36) results from the sum of aggregated detector elements, with a phase shift corresponding to the relative spacing between elements in the image domain.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.3.2 TDI OTF

This function applies the effects of time-delay-integration (TDI) to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The TDI can occur in either the x or y directions, and may exhibit a mismatch between the TDI clocking rate and the image motion rate. The OTF is given by:

$$H(u,v) = \operatorname{sinc}\left(\frac{dw}{f\beta}\right)\frac{1}{N_{tdi}\cdot N_{phases}}\sum_{j=0}^{N_{tdi}\cdot N_{phases}-1} e^{-i2\pi\frac{d(\beta-1)}{f\beta}jw} \tag{3-38}$$

where w is the spatial frequency in the TDI direction (u or v), d is the detector width in the TDI direction, f is the optical system focal length, N_{TDI} is the number of TDI stages, N_{phases} is the number of clock phases per transfer, and β is the ratio of the TDI clocking rate to image motion rate. The sinc function results due to the image motion over each TDI stage, while the summation results from the relative shifts between TDI stage images which arises due to the rate mismatch.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.3.3 CTE OTF

This function applies the effects of charge transfer inefficiency of a charge coupled device (CCD) detector array readout to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The OTF is given by [19]

$$H(u,v) = e^{-N_{phase}N_x(1-\varepsilon)\left[1-\cos\left(2\pi p_x u/f\right)\right]}\, e^{-N_{phase}N_y(1-\varepsilon)\left[1-\cos\left(2\pi p_y v/f\right)\right]} \tag{39}$$

where N_x and N_y are the number of transfers in x and y, p_x and p_y are the center-to-center detector spacings (pitch) in x and y, f is the optical system focal length, N_{phase} is the number of clock phases per transfer, and ε is the charge transfer efficiency (per phase).

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.3.4 Diffusion OTF

This function applies the effects of minority carrier diffusion in a charge coupled device (CCD) sensor to an image. It is characterized by an OTF and, therefore, operates in the spatial spectrum domain. The OTF is given by [20]

$$H(u,v)=\frac{1-\frac{e^{-\alpha ALD}}{1+\alpha AL(\rho)}}{1-\frac{e^{-\alpha ALD}}{1+\alpha ALO}} \tag{3-40}$$

where

$$AL(\rho)=\sqrt{\frac{1}{\frac{1}{ALO^2}+\left(\frac{2\pi\rho}{f}\right)^2}} \tag{3-41}$$

ρ is given by Eq. (3-4), α is the carrier spectral diffusion coefficient, ALD is the depletion layer width, ALO is the diffusion length, and f is the optical system focal length. Table 3-4 provides the carrier spectral diffusion coefficient for silicon at various wavelengths.

Spatial scaling (rad^{-1} units) attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the *Transform* tool.

3.3.5 Detection

This function performs a photoelectronic conversion of a focal plane irradiance image to a detected photoelectron image, including detector sensitivity, aggregation, dark current, and saturation. The conversion process is given by

$$N(x,y)=\frac{\lambda}{hc}\eta T_d N_{TDI} N_x N_y w_x w_y E(x,y)+\frac{J_d T_d N_{TDI} N_x N_y w_x w_y}{q} \tag{3-42}$$

where η is the quantum efficiency, T_d is the pixel dwell time, N_{TDI} is the number of TDI stages, N_x and N_y are the number of aggregated pixels in x and y, w_x and w_y are the

Table 3-4: Carrier Spectral Diffusion Coefficient for Silicon at Various Wavelengths

Wavelength (μm)	Coefficient (mm^{-1})
0.40	5000
0.45	1800
0.50	1000
0.55	650
0.60	450
0.65	300
0.70	200
0.75	150
0.80	95
0.85	60
0.90	35
0.95	20
1.00	10

detector widths in x and y, J_d is the dark current density, h is Planck's constant, c is the speed of light in a vacuum, and q is the electronic charge.

Spectral scaling attributes must be available for the input image, and the image data must be in irradiance (W/m^2 units). The output image data is in units of photoelectron counts.

3.3.6 Noise

This function adds a random noise process to an input image. Two types of noise are modeled. The first is signal dependent, Poisson distributed shot noise, which can be either enabled or disabled. The second is signal independent, additive noise that is assumed zero mean, gaussian distributed with a specified variance. The noise sources are assumed both jointly and pixel-to-pixel independent. The input and output image data must be in photoelectron units.

3.3.7 Quantization

This function quantizes a detected photoelectron image through a scalar roundoff process. Mathematically,

$$N_{out}(x,y)=\begin{cases} N_{\min} & N(x,y) < N_{\min} \\ N_{\min} + \dfrac{N_{\max}-N_{\min}}{2^b-1}\left\| (2^b-1)\dfrac{N(x,y)-N_{\min}}{N_{\max}-N_{\min}} + \dfrac{1}{2} \right\| & N_{\min} \le N(x,y) \le N_{\max} \\ N_{\max} & N(x,y) > N_{\max} \end{cases} \quad (3\text{-}43)$$

where N_{min} and N_{max} are the minimum and maximum input levels of the quantizer and b is the number of digitization levels (bits).

The input image data scaling is arbitrary, but must be consistent with the minimum and maximum levels.

3.3.8 Sampling

This function samples an input image on a rectilinear grid with arbitrary spacing. The sample locations in angular (radian) units are given by

$$x = x_{\min} + \left(i + \delta_x\right)\frac{p_x}{f} \tag{3-44}$$

$$y = y_{\min} + \left(j + \delta_y\right)\frac{p_y}{f} \tag{3-45}$$

where p_x and p_y are the detector center-to-center spacings in x and y, f is the optical system focal length, δ_x and δ_y are the user defined relative offset of the sampling grid in x and y (relative to pitch), x_{min} and y_{min} are the minimum field angles of the input image, and i and j are indices which range from zero to the maximum values such that the angular sample locations do not exceed the input image maximum field angles x_{max} and y_{max}. The resampling process is achieved using a bilinear interpolation of the input image data.

Spatial scaling (rad units) attributes must be available for the input image. The output image scaling attributes will vary depending on the sampling parameters, but will likely be different than the input.

3.3.9 Nonuniformity

This function applies the effects of random detector gain and/or offset nonuniformity to an input image. The nonuniformity can be characterized as an independent random variable between image samples, or the spatial correlation characteristics can be specified by an input autocorrelation image file. In both cases, realizations of gaussian distributed random gain and offset patterns are generated (independent random sample for each input image pixel). The gain pattern G(x,y) is unity mean with a standard deviation given by the gain nonuniformity and the offset pattern O(x,y) is zero mean with a standard deviation given by the offset nonuniformity times the maximum signal level.

When a spatial autocorrelation file is given, it is first resampled into a convolution kernel with an x and y sample spacing identical to the input image. This is performed by a bilinear interpolation. Next, it is normalized to exhibit unit variance. Finally, the gain and offset patterns are convolved by it to generate the desired spatial correlation characteristics.

Once the appropriate random gain and offset patterns are generated, the output image is computed by

$$I_{out}(x,y) = G(x,y)I_{in}(x,y) + 0(x,y) \tag{3-46}$$

The units of the input image are arbitrary, but must correspond to the specified maximum signal level for proper scaling of the offset nonuniformity. Spatial scaling attributes must exist for the input and autocorrelation files when an autocorrelation file is used. The units are arbitrary, but must be consistent.

3.4 DATA LINK

3.4.1 JPEG Compress

This function compresses an 8 or 12 bit image using the Joint Photographic Experts Group compression scheme [21] according to MIL-STD-188-198A [22]. The routine inputs a multiband IBSM image, an image quality factor, and a restart interval length. It compresses each band separately and returns both a compressed data stream and a multiband bits/block image. Compression is performed using code from the National Image Transmission Format Standard (NIFTS) JPEG Compression (NJC) Software Library (NJC) [23].

Bits/Block Image (BB Image)

The BB image for some arbitrary image has one element for each MCU in the image. This element contains the number of bits in the compressed representation of that MCU, and is used as an input for the *Finite Buffer* routine.

The BB image is derived by counting the number of bits in each restart interval and averaging this over the number of MCU's in a restart interval. The BB image is set equal to this average for the entire restart interval. Therefore the shorter the restart interval length, the more accurate the BB image. And (nearly) perfectly accurate BB images are produced if and only if the restart interval is one MCU long.

Minimum Coded Unit (MCU)

MCU is a JPEG term which stands for Minimum Coded Unit. To define it, start with any single-band image and divide it into 8x8 blocks which cover the whole image but yet do not overlap. Then each of these blocks is an MCU.

Image Quality Factor (IQF)

IQF is an NJC and IBSM term which denotes a number from 1 to 5 that controls the accuracy with which the image is reconstructed. Lower values mean that fewer bits are used in the compressed file. Unfortunately they also mean that the reconstructed image is cruder. This range of values (i.e. 1 to 5) was set by MIL-STD-188-198A, but is not part of the JPEG standard itself.

Restart Interval

This is a JPEG term which denotes a collection of one or more consecutive MCU's. After the compressed data for each restart interval, the compressor transmits a code. This may allow the receiver to regain synchronization when a bit error occurred in the compressed data for that restart interval. In other words, channel errors should be confined to the restart interval in which they occurred. So for optimal error protection we want the restart intervals as short as possible. The tradeoff is that each restart code takes about 20 bits to transmit, so the shorter the restart interval the more bits are spent on overhead.

To comply with MIL-STD-188-198A an image's restart interval must be no larger than the number of MCU's it takes to span the width of that image.

3.4.2 JPEG Decompress

This function is the inverse of *JPEG Compress*. It inputs a JPEG compressed data stream and returns an approximation to the original image.

3.4.3 TSVQ Compress/Decompress

Vector quantizers are often used to compress, with significant loss, sampled data such as images or sound. The class of vector quantizers has a particular subset whose

members, called Tree-Structured Vector Quantizers (TSVQ's), can achieve a given level of compression performance at a low complexity. [24] describes them in detail.

This function has two modes. The first reads an input image and a TSVQ codebook. It compresses the input using that codebook and mainly produces a sequence of indices that represents the input image. The second mode reads a sequence of indices and outputs the decompressed image.

The vector quantizer (VQ) described here is optimized to minimize the mean-squared error (MSE) between the reconstructed image and the original image. This is the customary way that Vqs are designed, mostly because it is conceptually the simplest distortion measure. Unfortunately it's bad to use MSE distortion if the reconstructed images will be primarily interpreted by humans, although this may work ok for machine interpretation. When human viewing is important, VQ's (of this type) are usually used as components in fancier data compressors such as wavelet coders.

This TSVQ program is based on programs from [25]. It is basically a straightforward TSVQ encoder and decoder, with the addition of restart markers. The restart marker scheme is similar to the one in the JPEG standard [21]. It starts by picking some natural number restart interval length (RI_len), then takes the stream of (possibly variable-length) indices and puts a marker after every RI_len of them. A marker consists of two bytes. The first is always 7F. The second is a number that cycles through the markers.

At the receiver, the data is parsed into segments delimited by markers. Each segment's marker determines the location in the image to begin placing the pixels corresponding to that segment's indices. This works perfectly if the markers never get corrupted. One complication is that 7F's, which indicate markers, can also appear naturally in the bits composing the indices. This is bypassed by stuffing a 00 byte after each natural occurrence of 7F. The receiver then needs to remember that 7F 00 needs to be translated to just 7F before the decompressor sees it.

Unfortunately, the markers can get corrupted. One coping strategy is to remember the index (i.e. the position in markers []) of the last decoded marker. Then look for the next marker whose index is greater. If its index isn't exactly one greater, then the receiver knows that an error has been made but, for simplicity, doesn't try to estimate the missing

marker's location. With this strategy, if an error occurs in a marker, then the entire segment corresponding to that marker is lost. Also, if an error occurs in the indices, then at most the rest of the segment is lost.

Bits/Unit Time File

Imagine a mesh whose interstices are 8x8 pixels square. Overlay this on the input image. The bits/(unit time) file has one 32-bit integer per interstice. Each of these integers is the number of bits emitted per unit time when the TSVQ is compressing the pixels in that interstice. It turns out that TSVQ's emit a constant number of bits per unit time so this file is not terribly exciting, but it's still sometimes needed.

3.4.4 Wavelet Compress/Decompress

Wavelet coding [24, 26, 27, 30] tries to approximate the compression performance of very high-dimensional vector quantization at a reasonable complexity. Suppose one wants to compress images that are N pixels long. The points in N-dimensional space that represent these images will typically be concentrated in a cloud that occupies only a small fraction of the volume of this space. The basic idea of wavelet encoding is to apply an N x N wavelet transform matrix to each of these points. This effectively orients the N coordinate axes with the cloud so that it's easier to describe the location of a particular image point. The description is still done with a vector quantizer (VQ), but the VQ has much less work because it happens that it can now consider many fewer than N dimensions at a time.

The wavelet decoder takes the indices that the VQ produced and reconstructs the image in the transformed coordinate system. Finally, it returns the image to the normal coordinate system by multiplying by the inverse wavelet transform. For a simple example of the advantages of a good coordinate system, consider the description of a point's location on a particular circle. In polar coordinates the description is economical because it requires just one number: the point's angle with respect to some reference ray. Conversely, in cartesian coordinates the description requires two numbers: the x and y positions.

This function has two modes: compress and decompress. It is executed based on a codebook from a training data set similar to that with TSVQ.

3.4.5 Finite Buffer

This program simulates the buffering of an IBSM format image. It inputs the image and its corresponding bits/block image (see the documentation for the JPEG glyphs for more information about these). The latter is produced automatically by IBSM compressor glyphs, or it may be produced manually for non-compressed images. The outputs of this function consist of the buffered data, which may be different from the original data if overflows and underflows occurred. It also produces a graph that represents how many bits were in the buffer at each time step.

This program keeps track of two kinds of events. If "Direction Toggle" is 0 then these are bit arrivals and block departures, otherwise they are bit departures and block arrivals. This program assumes that the time axis has been partitioned into intervals of length "Step size". It examines one interval at a time, simulates the events that happen in that interval and then, if requested, saves the number of bits remaining in the buffer at the end of the interval.

The bookkeeping it does is conceptually simple. The only twist is that there is no area of memory that serves as an actual buffer. Instead, the program keeps track of the number of bits in an imaginary buffer. If an incoming bit will fit in this buffer, then it is placed directly on the output since if there really were a buffer, then the bit would get there eventually anyway. This is simpler than first writing it into the buffer, and then writing it into the output.

The buffer model is based on a fixed rate source assumption; that is, there is a fixed number of seconds between arrivals of elements (either bits or blocks dependent on the direction toggle). Both buffer overflows and underflows can be simulated, and the buffer can be initialized to an initial level and/or delay prior to emptying.

3.4.6 Channel Errors

This program inputs an IBSM format image and simulates passing it through a noisy channel. It then outputs the resulting corrupted image. Any combination of the following channels may be used: binary symmetric (substitution) channels (BCS's), insertion channels, deletion channels, and burst error channels. Each bit passing through a

BSC has the same probability of being flipped. An insertion channel randomly inserts bits into the transmitted data stream, while a deletion channel randomly removes bits. Finally, a burst error channel corrupts sequences of consecutive bits.

This program is based on the idea that for each bit position in the file one of five events can happen: (no error, bit flip, bit insertion, bit deletion, burst error) = (e_0, e_1, e_2, e_3, e_4). Using this notation, each sequence of events is seen to be an element of {e_0, e_1, e_2, e_3, e_4}*. An obvious implementation is to use the inverse transformation method (described in, for example, [30]) at each bit position to map the event probabilities into a sequence from this set.

This program uses a faster alternative that is based on the equality of { e_0, e_1, e_2, e_3, e_4}* to (e_0,*[^e_0])*(e_0), which may be proved by induction on the length of a sequence. Writing each sequence in this new form is profitable because e_0 is usually much more probable than the other events. Therefore it's faster to use a geometric distribution to find the lengths of the e_0 runs, and to only use the inverse transformation method to determine the events that fall in-between these runs.

3.5 PERFORMANCE

3.5.1 Create Spectra

This function generates an eight-band IBSM file containing atmospheric and sensor spectral distributions. The bands contain, respectively, (0) atmospheric path transmission, (1) atmospheric path radiance, (2) irradiance at the target for a direct source, (3) diffuse downwelling radiance at the target, (4) target reflectance, (5) background reflectance, (6) sensor optical chain transmission, and (7) detector quantum efficiency. All are in single row format of "nwavl" columns. When the override parameters are not selected, the first four distributions (0-3) are computed via MODTRAN in reference to the ASCII path data file, the following two (4-5) from the ASCII reflectance file data, and the final two (6-7) from the ASCII sensor file data. In all cases, the spectral distributions are computed through a linear interpolation of the data, and can be overridden with the spectrally flat override parameters. The minimum and maximum x scaling attributes contain the wavelength minimum and maximum in meters. The minimum y, maximum y, center wavelengths, and spectral bandwidths attributes should be ignored. This generated IBSM file is of a form suitable for input to the *Sensor Performance* function.

3.5.2 Create MTF

This function generates a one-dimensional Modulation Transfer Function (MTF) based on a real-valued two-dimensional spatial spectrum file representing the MTF. The generated MTF can lie along either the X or Y axes of the input data, and is placed as a single row in an IBSM output file. The frequency scale (min/max) can be either the same as the input (automatic) or overridden (fixed). In the latter case, the data is linearly interpolated to the specified grid. This generated IBSM file is of a form suitable for input to the *Sensor Performance* function.

3.5.3 Sensor Performance

This function computes metrics characterizing sensor performance (including SNR, GRD, and NIIRS) based on a set of sensor parameters; target, background, atmosphere, and sensor spectral distributions using the *Create Spectra* function; and x and y axis MTF functions using the *Create MTF* function. The metric is written to an ASCII file along with an arbitrary independent variable. Depending on the "update" parameter, the file is initialized before writing or the data is appended to an existing file. In the long form output data format, the full input parameter set is written to the file upon initialization (update = 0), and the metric data is augmented with additional intermediate computation results. The short form output data format results in a two column ASCII file containing the independent variable and computed metric for each update execution, which is useful for generating parametric plot data.

The Signal to Noise Ratio (SNR) metric refers to the target to background detected signal difference relative to the noise level. It is computed by:

$$SNR = \frac{N_{HI} - N_{LOW}}{\sqrt{N_{HI} + \frac{J_d N_x N_y w_x w_y T_d}{q} + {\sigma_n}^2 + \frac{1}{12}\left[\frac{N_{\max} - N_{\min}}{2^b - 1}\right]^2}} \tag{3-47}$$

where

$$N_{HI} = \max\left(N_{tgt}, N_{bkg}\right) \tag{3-48}$$

$$N_{LOW} = \min\left(N_{tgt}, N_{bkg}\right) \tag{3-49}$$

$$N_{tgt} = \frac{\pi T_d N_{TDI} N_x N_y w_x w_y}{4hc(f/\#)^2} \left\{ L_{stray} + \int_{\lambda_{\min}}^{\lambda_{\max}} \lambda \eta(\lambda) \left[\begin{array}{c} \varepsilon_{opt} L_{BB}\left(\lambda, T_{opt}\right) + \tau_{opt}(\lambda)\tau_{atm}(\lambda) L_{tgt}(\lambda) \\ + \tau_{opt}(\lambda) L_{path}(\lambda) \end{array} \right] d\lambda \right\} \tag{3-50}$$

$$N_{bkg} = \frac{\pi T_d N_{TDI} N_x N_y w_x w_y}{4hc(f/\#)^2} \left\{ L_{stray} + \int_{\lambda_{\min}}^{\lambda_{\max}} \lambda \eta(\lambda) \left[\begin{array}{c} \varepsilon_{opt} L_{BB}\left(\lambda, T_{opt}\right) + \tau_{opt}(\lambda)\tau_{atm}(\lambda) L_{bkg}(\lambda) \\ + \tau_{opt}(\lambda) L_{path}(\lambda) \end{array} \right] d\lambda \right\} \tag{3-51}$$

$$L_{tgt}(\lambda) = \left[1 - \rho_{tgt}(\lambda)\right] L_{BB}\left(\lambda, T_{tgt}\right) + \rho_{tgt}(\lambda) \left[\frac{E_{direct}(\lambda)\cos\xi}{\pi} + L_{diffuse}(\lambda) \right] \tag{3-52}$$

$$L_{bkg}(\lambda) = \left[1 - \rho_{bkg}(\lambda)\right] L_{BB}\left(\lambda, T_{bkg}\right) + \rho_{bkg}(\lambda) \left[\frac{E_{direct}(\lambda)\cos\xi}{\pi} + L_{diffuse}(\lambda) \right] \tag{3-53}$$

and $L_{BB}(\lambda,T)$ is the blackbody function given in Eq. (3-35). The parameters used in Eqs. (3-47) to (3-53) are defined in Table 3-5.

The SNR computation also includes saturation effects and optionally automatic gain control. Saturation is incorporated by limiting N_{HI} and N_{LOW} to within the minimum to maximum range. Automatic gain control is performed by reducing both N_{HI} and N_{LOW} by a factor such that N_{HI} equals a specified AGC level. If N_{HI} is less than this AGC level initially, no scaling is performed.

The Ground Resolved Distance (GRD) metric is computed orthogonal to the sensor line-of-sight axis. Initially, it is computed in both the x and y axes by

$$GRD_{x,y} = \max\left[\frac{R}{u_r}, \frac{2p_{x,y}R}{f} \right] \tag{3-54}$$

Table 3-5: SNR Performance Parameter Definitions

$L_{diffuse}(\lambda)$	Diffuse downwelling radiance (W/m^3sr)
$E_{direct}(\lambda)$	Direct source irradiance (W/m^3)
ξ	Source zenith angle (deg)
$\rho_{tgt}(\lambda)$, $\rho_{bkg}(\lambda)$	Target, background reflectance
T_{tgt}, T_{bkg}	Target, background temperature (Kelvin)
$L_{tgt}(\lambda)$, $L_{bkg}(\lambda)$	Target, background radiance (W/m^3sr)
$L_{path}(\lambda)$	Atmospheric path radiance (W/m^3sr)
$\tau_{atm}(\lambda)$	Atmospheric path transmission
$\tau_{opt}(\lambda)$	Optical train transmission
ε_{opt}	Effective optics emissivity
T_{opt}	Effective optics temperature (Kelvin)
$\eta(\lambda)$	Detector quantum efficiency
L_{stray}	In-band stray radiance (W/m^3sr)
T_d	Dwell time (msec)
N_{TDI}	Time - delay - integrate stages
f/#	Optics F-number
N_x, N_y	Aggregation in x and y
W_x, W_y	Detector width in x and y (μm)
N_{min}, N_{max}	Quantizer minimum and maximum
b	Quantizer digitization levels
σ_n	Detector rms noise (electrons)
J_d	Detector dark current density ($\mu A/cm^2$)
N_{tgt}, N_{bkg}	Target, background signal level (electrons)

where R is the sensor to target slant range, p_x and p_y are the detector center-to-center spacings in x and y, f is the optical system focal length, and u_r is the highest spatial frequency to satisfy

$$MTF_{x,y}(u_r) \geq \frac{3}{SNR} \tag{3-55}$$

The final GRD is given as the geometric mean (square root of the product) of the GRD in x and y.

Image quality is specified by a National Image Interpretability Rating Scale (NIIRS) value. This is computed from the SNR and modulation transfer functions in x and y by the Generalized Image Quality Equation [29]:

$$Q = 11.81 + 3.32 \log_{10} (\text{RER} / \text{GSD}) - 1.48\text{H} - \frac{\text{G}}{\text{SNR}} \tag{3-56}$$

where

$$H = \sqrt{H_x H_y} \tag{3-57}$$

$$RER = \sqrt{RER_x RER_y} \tag{3-58}$$

$$GSD = \sqrt{GSD_x GSD_y} \tag{3-59}$$

$$H_{x,y} = \begin{cases} \max ER_{x,y}(\xi = 1.0 \; to \; 3.0) & monotonic \\ ER_{x,y}(1.25) & not \; monotonic \end{cases} \tag{3-60}$$

$$RER_{x,y} = ER_{x,y}(0.5) - ER_{x,y}(-0.5) \tag{3-61}$$

$$GSD_{x,y} = \frac{p_{x,y} R}{f} \tag{3-62}$$

$$ER_{x,y}(\xi) = 0.5 + \frac{1}{\pi} \int_0^{D/\lambda_{\min}} \frac{MTF_{x,y}(w)}{w} \sin\left(\frac{2\pi w N_{x,y} p_{x,y} \xi}{f} \right) dw \tag{3-63}$$

The parameters used in Eqs. (3-56) to (3-63) are given in Table 3-6. NIIRS is generally specified in reflective bands for a target reflectance of 0.15 and background reflectance of 0.07, and in thermal bands for blackbody radiance with a target temperature of 302K and background temperature of 300K.

3.5.4 Exposure Control

This function performs exposure control by determining the pixel dwell time, TDI stages, and aggregation levels which place the maximum of the target and background signal levels between specified minimum and maximum levels. For the signal computation, sensor spectral distributions using the *Create Spectra* function are used. The signal computation is given by Equations (3-50) to (3-53).

Initially, the signal level is computed for a user defined desired dwell time, the maximum allowable TDI, and the minimum allowable aggregation. If the maximum of the target and background signal level (called initial signal) falls between the minimum to maximum signal window, the initial sensor configuration values are unchanged.

If the initial signal exceeds the maximum (saturation condition), first the TDI stages are decremented by steps. Once the adjusted signal level falls below the maximum, the process stops with a new number of TDI stages having been established. If the saturation condition cannot be resolved with the minimum allowable TDI, the dwell time is reduced to place the adjusted signal to the maximum level with the minimum allowable TDI stages.

If the initial signal is less than the minimum level (low signal condition), first the aggregation levels are incremented (alternately x and y) by steps. Once the adjusted signal level falls above the minimum, the process stops with the new aggregation levels having been established. If the low signal condition cannot be resolved with the maximum aggregation, the dwell time is increased to place the adjusted signal to the minimum level with the maximum aggregation.

The output consists of the sensor configuration values, which are placed in a five-band floating-point VIFF file (one value per band) in the following order: (0) pixel dwell time in msec, (1) TDI stages, (2) X aggregation, (3) Y aggregation, (4) adjusted signal level. The values can be extracted to global variables using the *Print Value* function.

Table 3-6: NIIRS Performance Parameter Definitions

ξ	Pixel position
$p_{x,y}$	Detector pixel pitch (μm)
$N_{x,y}$	Aggregation in x and y
w	Spatial frequency (rad^{-1}) in x or y
f	Optical system focal length (cm)
D	Aperture diameter (cm)
λ_{min}	Minimum wavelength (μm)
$ER_{x,y}(\xi)$	Line response function in x and y
R	Target to sensor slant range (km)
$GSD_{x,y}$	Ground sample distance in x and y
$RER_{x,y}$	Relative edge response in x and y
$H_{x,y}$	Edge height overshoot in x and y
SNR	Signal to noise ratio
G	SNR gain due to image processing
RER	Relative edge response (mean)
H	Edge height overshoot (mean)
GSD	Ground sample distance (mean)
Q	Image quality (NIIRS)

3.5.5 Image Quality

This function computes image quality metrics including line response functions, GSD, SNR, and NIIRS from an image containing test patterns and/or edge features. The user specifies the locations, in angular (mrad) units, of horizontal and vertical edges as well as ideally uniform image regions for noise estimation.

First, the function extracts, centers, and normalizes the vertical and horizontal edge response functions. The image is assumed to exhibit the proper sampling characteristics such that these will correspond to $ER_{x,y}(\zeta)$ in Eq. (3-63) where ζ corresponds to image pixels. Next, the mean and variance over designated target and background regions is computed. Finally, the image quality metrics are computed via Equations (3-56) through (3-62) along with

$$SNR = \frac{\Delta\mu}{\sigma_{max}} \tag{3-64}$$

where $\Delta\mu$ is the difference between the mean signal levels and σ_{max} is the standard deviation associated with the region exhibiting a higher mean signal.

Based on user options, the function output consists of the two edge response functions in two column ASCII format and an ASCII metric summary file. Spatial scaling (rad units) attributes must exist for the input image, and the specified edge and region locations must fall within the angular range of the image.

3.6 PROCESSING

3.6.1 Image Interlace

This function interlaces up to four images in a user-specified direction. The direction can be either along the x axis (width), y axis (height), or in bands (element). The attribute information is also appropriately adjusted, although care must be taken such that the input attribute information is appropriate. For example, nonsensible output scaling attributes will occur if the input scaling data was not representative of interleaved images. The interlaced images must be of the same size and data type.

3.6.2 Aggregation

This function sums image elements in width, height, and/or bands to a user specified aggregation level. By virtue of the aggregation, the output image is smaller in each dimension by the aggregation factor. The residual image, when the size is not an integer multiple of the aggregation factor, is truncated. If the projection option is selected for a particular dimension, the aggregation factor is overridden by the dimension size. Note that overflows may occur in the summation process.

3.7 UTILITIES

3.7.1 VIFF to IBSM

This function adds attribute information to a VIFF file to create an IBSM format output. The spatial scaling and spectral band data can either be entered from the screen (up to four bands), or input from an ASCII file. The ASCII file format, displayed in Table 3-7, corresponds to that generated using the *List IBSM File Data* function.

3.7.2 IBSM Info

This function generates an ASCII file containing information extracted from the IBSM header and attributes. This includes data type, image size, number of bands, spatial scaling, and the center wavelength and spectral bandwidth for each band. The file format, displayed in Table 3-7, is appropriate for input to the *VIFF to IBSM* utility.

3.7.3 Transform

This function performs conversions between the image and spatial spectrum domains through a discrete Fourier transform. The forward and reverse transformations are given by [30]

$$G(u,v) = \sum_{j=0}^{N_x-1} \sum_{k=0}^{N_y-1} I(x_j, y_k) e^{i2\pi(ux_j + vy_k)} \tag{3-65}$$

$$I(x,y) = \frac{1}{N_x N_y} \sum_{j=0}^{N_x-1} \sum_{k=0}^{N_y-1} G(u_j, v_k) e^{-i2\pi(xu_j + yv_k)} \tag{3-66}$$

Table 3-7: IBSM Header File Format

File Name: *fname*

File Type: IBSM Khoros image file format

VIFF Information:

Data Type:	*data-type*
X_Pixels:	*x_pixels*
Y_Pixels:	*y_pixels*
Bands:	*bands*

IBSM Information:

X_Min:	*x_min*
X_Max:	*x_max*
Y_Min:	*y_min*
Y_Max:	*y_max*

Wavelengths	Bandwidths
wavelength (1)	*bandwidth (1)*
•	
•	
•	
wavelength (bands)	*bandwidth (bands)*

where I(x,y) is an image, G(u,v) is the corresponding spatial spectrum, (x,y) are angular coordinates (rad), (y,v) are angular spatial frequency coordinates (rad^{-1}), and N_x and N_y are the transform sizes in x and y.

For computational efficiency, the transforms are performed using a Fast Fourier Transform (FFT) algorithm. The use of this algorithm places some constraints on the transform size sampling. The transform sizes must be a power of two in x and y. If the image or spectrum to be transformed is not an exact power of two in size, a larger transform size must be used. The input image or spectrum is automatically zero padded to this size. The sampling of the transform is also automatically computed based on the sample spacing of the input. Specifically,

$$u_{\min} = \frac{-N_x}{2} \frac{1}{x_{\max} - x_{\min}} \tag{3-67}$$

$$u_{\max} = \left(\frac{N_x}{2}\right) \frac{1}{x_{\max} - x_{\min}} \tag{3-68}$$

$$v_{\min} = \frac{-N_y}{2} \frac{1}{y_{\max} - y_{\min}} \tag{3-69}$$

$$v_{\max} = \left(\frac{N_y}{2} - 1\right) \frac{1}{y_{\max} - y_{\min}} \tag{3-70}$$

for the forward transform case where the image scale factors (x_{min}, x_{max}, y_{min}, y_{max}) correspond to after the zero padding operation. Note that the transform is always essentially centered, which is accomplished by quadrant swapping prior to and after the FFT.

Spatial scaling attributes must be available for the input image, although the image data units are arbitrary. Both complex and real input and output data types are supported.

3.7.4 Create Plot Data

This function extracts a row or column of an IBSM file and outputs it along with spatial scaling data (extracted from the scaling attributes) into two column ASCII format for plotting.

3.7.5 Create Radiance Image

This function converts a ground reflectance and, optionally, temperature distribution into a radiance image, including both reflected and radiated components. The reflected component uses only the reflectance image input (assumed Lambertian) with direct and diffuse illumination computed via MODTRAN in reference to the ASCII path data file. This file can be generated using the *MODTRAN Input* tool. It is important that the wavelength limits specified in the MODTRAN path data file cover each spectral band specified in the input image header. The direct, diffuse, or both illumination values from MODTRAN can be overridden. The radiated component is computed using both the reflectance and temperature (Kelvin units) images, and is ignored when the latter does not exist.

Mathematically, the output image is given by

$$I(x,y) = \rho(x,y) \int_{\lambda_0-\Delta\lambda/2}^{\lambda_0+\Delta\lambda/2} \left[\frac{E_{direct}(\lambda)\cos\phi_d}{\pi} + L_{diffuse}(\lambda) \right] d\lambda + \left[1-\rho(x,y)\right] \int_{\lambda_0-\Delta\lambda/2}^{\lambda_0+\Delta\lambda/2} L_{BB}\left[\lambda, T(x,y)\right] d\lambda \tag{3-71}$$

where $\rho(x,y)$ is the reflectance image, $T(x,y)$ is the temperature image, $E_{direct}(\lambda)$ is the directional spectral illumination, ϕ_d is the zenith angle of the directional source, $L_{diffuse}(\lambda)$ is the diffuse downwelling spectral radiance, and $\Delta\lambda$ is the spectral bandwidth for the band of interest.

Spectral band attributes must be available in the reflectance input image, and the data must be in reflectance units. The temperature input file data must be in units of absolute temperature (Kelvin). The output image data is in radiance (W/m^2sr) units.

3.7.6 Ground/Angle Transform

This function performs a coordinate transformation of a radiance image between the horizontal ground plane (meter units) and line-of-sight angular (radian units) coordinates. Only the image scaling attribute values are altered based on the viewing geometry. The defined azimuth axis is the non-squinted axis (normal to the line-of-sight and vertical axes). The scaling of the squinted axis is based on a linear projection, which occurs under the assumption that the ground field-of-view in the squinted direction is much smaller than the slant range. No image distortion or perspective effects are modeled.

If the input image is scaled in meter units in the viewing plane (orthogonal to line-of-sight) rather than the horizontal plane, this utility can be used with a zenith angle of zero to convert to angular units.

Spatial scaling attributes in meter units must exist for the input image. The spectral band attributes and image data are arbitrary.

3.7.7 Load Variables

This function loads a set of global variable definitions from an ASCII file. The file format parallels the "Global Variables" input form, and include simple definitions of variables as constants, expressions, and comments. Comments are denoted by a "#" at the start of a line.

3.7.8 Xgraph Plot

This function provides a user interface to the Xgraph plotting application, and allows up to four plots per graph with preformatted titles, labels, and axes ranges. The data is input as separate, two-column ASCII files. Xgraph is executed from a system call.

4.0 USAGE OVERVIEW

In this section, information is provided in an effort to assist the user in developing and executing sensor models based on the IBSM toolbox. Detailed information on the use of Khoros and, specifically, Cantata is available elsewhere [31] and is not repeated here. However, brief descriptions of some commonly used features are given.

4.1 FILE FORMATS

Image data, as well as other data at times, is passed between IBSM modules and other Cantata glyphs (Khoros term for functional module) by files in the Khoros Visualization/Image File Format (VIFF). The Khoros Image File Format is organized as 1024 bytes of header followed by map(s), location data and then image data. The header also contains information identifying the header format used. This file format or data structure has evolved from originally supporting only *images* to supporting multi-dimensional data. The multidimensional aspect is used to accommodate multiband imagery.

For use with IBSM, specific header information relating to spatial scaling and spectral band characteristics is added as *attributes* to the VIFF files. Attribute data is merely arbitrary appended data to a VIFF file. Many of the IBSM modules require this information to be present for correct operation. This extended VIFF file format is referred to as IBSM file format. The *VIFF to IBSM* Utility is used to add this attribute information to a VIFF file. Since the attribute data is essentially extraneous in the VIFF file format, IBSM file formats are compatible with all the standard Khoros functions and, in most cases, the attribute data is passed through undisturbed.

There are four basic types of IBSM attribute information. The first is a image type flag: IBSM_FLAG. This denotes that the VIFF file is of IBSM format (IBSM_FLAG = 85). The second is spatial scaling information: IBSM_SCALE. This contains four values which, in order, specify X_{min}, X_{max}, Y_{min}, and Y_{max}. X_{min} specifies the first column of the image, and Y_{min} the first row. The third is spectral band information: IBSM_WAVELENGTH and IBSM_BANDWIDTH. Each is an array of values containing the spectral band centers and bandwidths, respectively, for each band of image data in meter units. The fourth is data compression attributes: IBSM_COMPRESSION_TYPE, IBSM_COMPRESSION_BAND_SIZE, and IBSM_ORIGINAL_SIZE. Table 4-1

Table 4-1: IBSM_COMPRESSION_TYPE Attribute Values

Value	Compression Type
1	JPEG (8 bit)
2	JPEG (12 bit)
3	VQ
4	DPCM
5	Wavelet

summarizes the IBSM_COMPRESSION_TYPE attribute values. IBSM_COMPRESSION_BAND_SIZE is given in bytes and IBSM_ORIGINAL_SIZE in width and height elements.

Image data is stored as part of the VIFF 5-dimensional data object. The multiband IBSM imagery utilize the WIDTH dimension for the x-axis, HEIGHT dimension for the y-axis, and ELEMENT dimension for bands.

Table 4-2 summarizes the assumed input and output attribute data (not including compression) for correct operation of each IBSM function. Table 4-3 summarizes the input and output file types and assumed units (if applicable) for each.

4.2 MODEL GENERATION

Cantata is a graphical programming environment for generating, in the case of IBSM, sensor evaluation models. Each of the IBSM functions, along with over a hundred standard data processing and visualization programs in the Khoros system, are represented by icons (called glyphs). To create a model, the user selects the desired glyphs (and control structures, as needed), places them in the Cantata workspace, and connects them to indicate the flow of the model. Such a model is called a workspace. Workspaces can then be executed, saved, and restored to be used again or modified later. The save and restore features are implemented by selecting the *Workspace* button.

4.3 DATA AND IMAGE VIEWING

Three primary data and image viewing capabilities exist under Cantata, all of which are selected under the *Output* button. Information contained in ASCII files can be displayed using the *ASCII File Viewer* contained under the *Information* heading. Imagery can be displayed in a variety of ways, all contained under the *Display Image* heading. Finally, data contained in ASCII or VIFF files can be plotted using the *Xprism 2* Utility under the *Plot Data* heading or using the *Xgraph Plot* IBSM utility. Plots are placed on the display in a default mode, but can be altered by the user and output to a printer or file.

Table 4-2: Assumed IBSM Attribute Information

TOOLBOX	FUNCTION	INPUT ATTRIBUTES			OUTPUT ATTRIBUTES		
		IBSM_SCALE	IBSM_WAVELENGTH	IBSM_BANDWIDTH	IBSM_SCALE	IBSM_WAVELENGTH	IBSM_BANDWIDTH
ATMOSPHERE	MODTRAN Input	none	none	none	none	none	none
	Path Trans/Rad	arbitrary	meters	meters	passed	meters	meters
	Turbulence OTF	rad^{-1}	meters	arbitrary	rad^{-1}	meters	passed
	Aero-Optic OTF	rad^{-1}	meters	arbitrary	rad^{-1}	meters	passed
OPTICS	Diffraction OTF	rad^{-1}	meters	arbitrary	rad-1	meters	passed
	Defocus OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Jitter OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Drift OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Wavefront OTF	rad^{-1}	meters	arbitrary	rad^{-1}	meters	passed
	User OTF (1-D)	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	User OTF (2-D)	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Radiometry	arbitrary	meters	meters	passed	meters	meters
DETECTOR	Detector Size OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	TDI OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	CTE OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Diffusion OTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Detection	arbitrary	meters	arbitrary	passed	passed	passed
	Noise	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Quantization	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Sampling	rad	arbitrary	arbitrary	rad	passed	passed
	Nonuniformity	arbitrary	arbitrary	arbitrary	passed	passed	passed
DATA LINK	JPEG Compress	arbitrary	arbitrary	arbitrary	passed	passed	passed
	JPEG Decompress	arbitrary	arbitrary	arbitrary	passed	passed	passed
	TSVQ Compress/Decompress	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Wavelet Comp/Decomp	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Finite Buffer	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Channel Errors	arbitrary	arbitrary	arbitrary	passed	passed	passed
PERFORMANCE	Create Spectra	none	none	none	meters	zero	zero
	Create MTF	rad^{-1}	arbitrary	arbitrary	rad^{-1}	passed	passed
	Sensor Performance	meters	arbitrary	arbitrary	none	none	none
	Exposure Control	meters	arbitrary	arbitrary	none	none	none
	Image Quality	rad	arbitrary	arbitrary	none	none	none
PROCESSING	Image Interlace	arbitrary	arbitrary	arbitrary	passed	passed	passed
	Aggregation	arbitrary	arbitrary	arbitrary	passed	passed	passed

Table 4-2: Assumed IBSM Attribute Information (Continued)

TOOLBOX	FUNCTION	INPUT ATTRIBUTES			OUTPUT ATTRIBUTES		
		IBSM_SCALE	IBSM_WAVELENGTH	IBSM_BANDWIDTH	IBSM_SCALE	IBSM_WAVELENGTH	IBSM_BANDWIDTH
UTILITIES	VIFF to IBSM	none	none	none	arbitrary (rad)	meters	meters
	List IBSM File Data	arbitrary	meters	meters	none	none	none
	Transform	arbitrary	arbitrary	arbitrary	inverted	passed	passed
	Create Plot Data	arbitrary (reqd.)	arbitrary	arbitrary	none	none	none
	Create Radiance Image	arbitrary	meters	meters	passed	meters	meters
	Ground/Angle Transform	meters	arbitrary	arbitrary	rad	passed	passed
	Image Interlace	arbitrary	arbitrary	arbitrary	changed	passed	passed
	Load Variables	none	none	none	none	none	none
	Xgraph Plot	none	none	none	none	none	none

Table 4-3: Input and Output File Types and Units

Toolbox	Function	Input File Type	Input File Data Type	Input File Units	Output File Type	Output File Data Type	Output File Units
Atmosphere	Modtran Input	None	None	None	ASCII	ASCII	None
	Path Trans/Rad	IBSM and ASCII	Real	Arbitrary	IBSM	Float	W/m^2sr
	Turbulence OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Aero-Optic OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
Optics	Diffraction OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Defocus	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Jitter OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Drift OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Wavefront OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	User OTF (1-D)	IBSM and ASCII	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	User OTF (2-D)	IBSM (2)	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Radiometry	IBSM	Float	W/m^2sr	IBSM	Float	W/m^2sr
Detector	Detector Size OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	TDI OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	CTE OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Diffusion OTF	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Detection	IBSM	Float	W/m^2sr	IBSM	Float	electrons
	Noise	IBSM	Float	electrons	IBSM	Float	electrons
	Quantization	IBSM	Float	Arbitrary	IBSM	Float	Arbitrary
	Sampling	IBSM	Float	Arbitrary	IBSM	Float	Arbitrary
	Nonuniformity	IBSM	Float	Arbitrary	IBSM	Float	Arbitrary

Table 4-3: Input and Output File Types and Units (Continued)

Toolbox	Function	Input File Type	Input File Data Type	Input File Units	Output File Type	Output File Data Type	Output File Units
Performance	Create Spectra	ASCII	ASCII	None	IBSM	Float	Various
	Create MTF	IBSM	Float	Arbitrary	IBSM	Float	MTF
	Sensor Performance	IBSM	Float	Various	ASCII	ASCII	None
	Exposure Control	IBSM	Float	Various	VIFF	Float	msec
	Image Quality	IBSM	Float	Arbitrary	ASCII (optionally 3)	ASCII	None
Utilities	VIFF to IBSM	VIFF	Arbitrary	Arbitrary	IBSM	Arbitrary	Arbitrary
	IBSM Infor	IBSM	Arbitrary	Arbitrary	ASCII	ASCII	None
	Transform	IBSM	Float, complex	Arbitrary	IBSM	Float, complex	Arbitrary
	Create Plot Data	IBSM	Float	Arbitrary	ASCII	ASCII	None
	Create Radiance Image	IBSM (optionally 2)	Float	Refl.(Kelvin)	IBSM	Float	W/m^3sr
	Ground/Angle Transform	IBSM	Arbitrary	Arbitrary	IBSM	Arbitrary	Arbitrary
	Image Interlace	IBSM	Arbitrary	Arbitrary	IBSM	Arbitrary	Arbitrary
	Load Variables	ASCII	ASCII	None	None	None	None
	Xgraph Plot	ASCII (up to 4)	ASCII	None	None	None	None
Processing	Image Interlace	IBSM	Arbitrary	Arbitrary	IBSM	Arbitrary	Arbitrary
	Aggregation	IBSM	Arbitrary	Arbitrary	IBSM	Arbitrary	Arbitrary
Data Link	JPEG Compress	IBSM	Byte, short	None	IBSM	Byte	None
	JPEG Decompress	IBSM	Byte	None	IBSM	Byte, short	None
	TSVQ Compress/Decompress	IBSM	Byte, short	None	IBSM	Byte	None
	Wavelet Comp/Decomp	IBSM	Byte, short	None	IBSM	Byte, short	None
	Finite Buffer	IBSM	Byte	None	IBSM	Byte	None
	Channel Errors	IBSM	Byte	None	IBSM	Byte	None

4.4 STANDARD KHOROS TOOLBOXES

In addition to the IBSM functions incorporated under Khoros, the user can utilize a broad range of standard utilities for signal and image processing by selecting glyphs from the standard toolboxes. A brief overview is given here.

4.4.1 Arithmetic

This toolbox contains routines for a range of mathematical image operations, including unary operations (scaling, offset, absolute value, square root, etc.), binary operations (addition, subtraction, etc.), logical operations (And, Or, etc.), and linear transforms.

4.4.2 Control

This toolbox contains routines for looping and conditional constructs. The control functions are useful particularly for looping (e.g., computing sensor performance as a function of some parameter) and conditional branches. The generic command allows the user to insert an arbitrary UNIX command line program anywhere within a workspace. Loops and procedures (groups of glyphs combined into one) contain all included glyphs as a sub-workspace.

4.4.3 Data Manipulation

This toolbox contains a range of image processing functions including analysis (statistics), data conversion, transforms, frequency filters, histograms, geometric operations (expanding, shrinking, transposing, etc.), and subregion operations (cropping, inserting, etc.)

4.4.4 Geometry

This toolbox contains routines for operating with geometrical constructs.

4.4.5 Image Processing

This toolbox contains routines for data compression, geometrical operators, and image filtering.

4.4.6 Input/Output

This toolbox contains utilities for inserting supplied or user defined image and signal data files into a workspace, or for creating images or signals, and converting file formats..

4.4.7 Matrix

This toolbox contains routines for matrix manipulation.

4.4.8 Program Utilities

This toolbox contains generic interfaces, and command and comment icons.

4.4.9 Visualization

This toolbox contains routines for image and data capture and display.

4.5 GLOBAL VARIABLES

The *Global Variables* button is used to define variables and evaluate expressions within the Cantata environment. Once defined, variables can be used in the place of integer, float, and double arguments (GUI entries) of glyphs. Also, expressions can be defined to establish interrelationships between variables (and, therefore, model parameters) which can be dynamic in the course of a workspace or model execution.

For example, consider a sensor performance model which loops over some independent variable. It is conceivable that the IBSM modules in each iteration of the loop change dependent on the loop variable. This relationship is established by representing the IBSM module input as a variable and defining the variable by some functional dependence to the loop variable. Valid expressions for this functional dependence include variables,

standard arithmetic operators and logicals, as well as predefined constants and functions. Any string of alphanumeric characters, beginning with a letter, may be used as a variable.

In addition to defining functional dependencies between global variables and defining the glyph operation in terms of such variables, it is also possible to define a dependence of a global variable based on the result of a glyph operation. This can be accomplished using the *Print Value* utility under the *Data Manip / Analysis and Information* heading, which allows the user to assign any element of a VIFF file to a global variable.

4.6 ON-LINE DOCUMENTATION

On-line documentation (or help) is provided by clicking on *help* buttons. The on-line help pages that are displayed by a particular help button are appropriate to the location of the graphical user interface on which the help button is found. Since the Cantata master form is at the highest level of the Cantata GUI hierarchy, the help button at the upper right hand corner of the Cantata master form displays by default the introduction to Cantata. Notice the button at the upper left hand corner of the on-line help display labeled, "More Help Pages". This submenu button allows you to display related help pages; in this case, all the other help pages for Cantata, organized not alphabetically but in the order in which they appear in the manual.

Help buttons found on each subform will display help pages which describe the purpose and operation of that particular subform. Again, any other related help pages may be accessed by using the submenu button labeled, "More Help Pages."

The on-line documentation actually places the man pages for a particular function into a user screen. In addition to a description of the function and usage, a description of the model inputs is given. Since each function is executable from a UNIX command line, each input is associated with a unique command line switch. Therefore, the various inputs in the documentation are indicated by these switches. The association with GUI descriptions is not explicitly given, but is probably obvious in most cases.

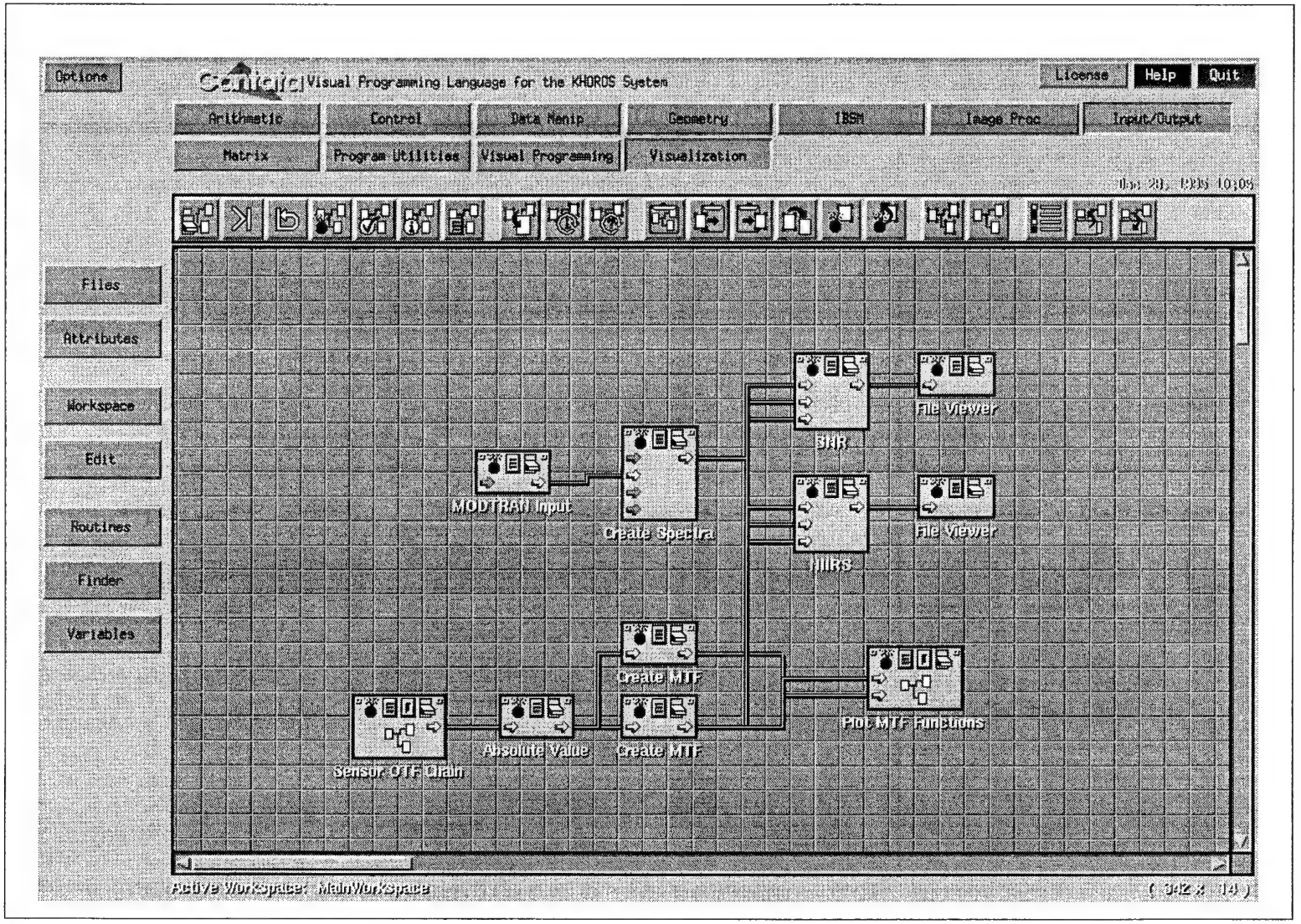

Figure 5-1: Workspace Corresponding to Test Case

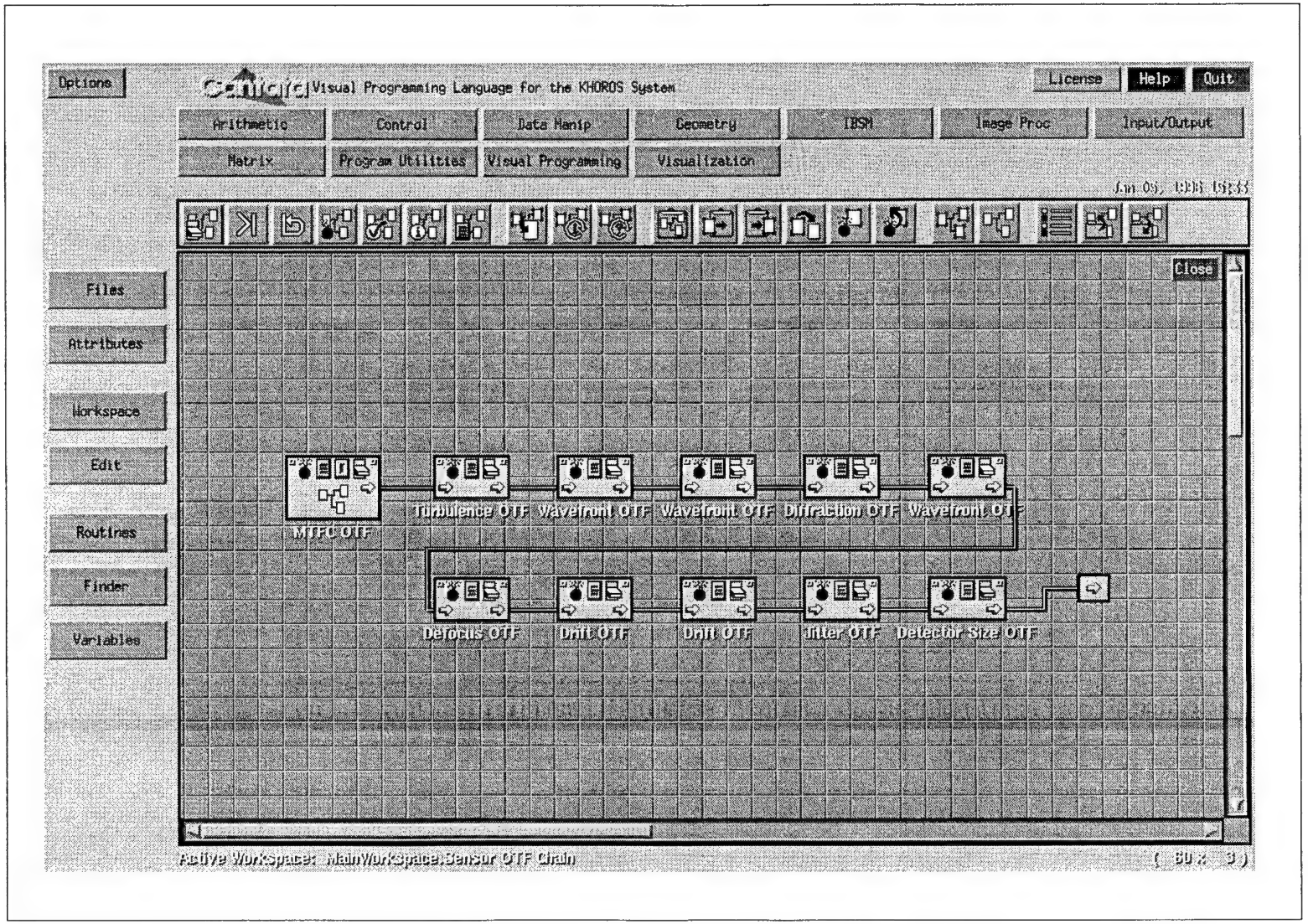

Figure 5-2: Sensor OTF Chain Procedure

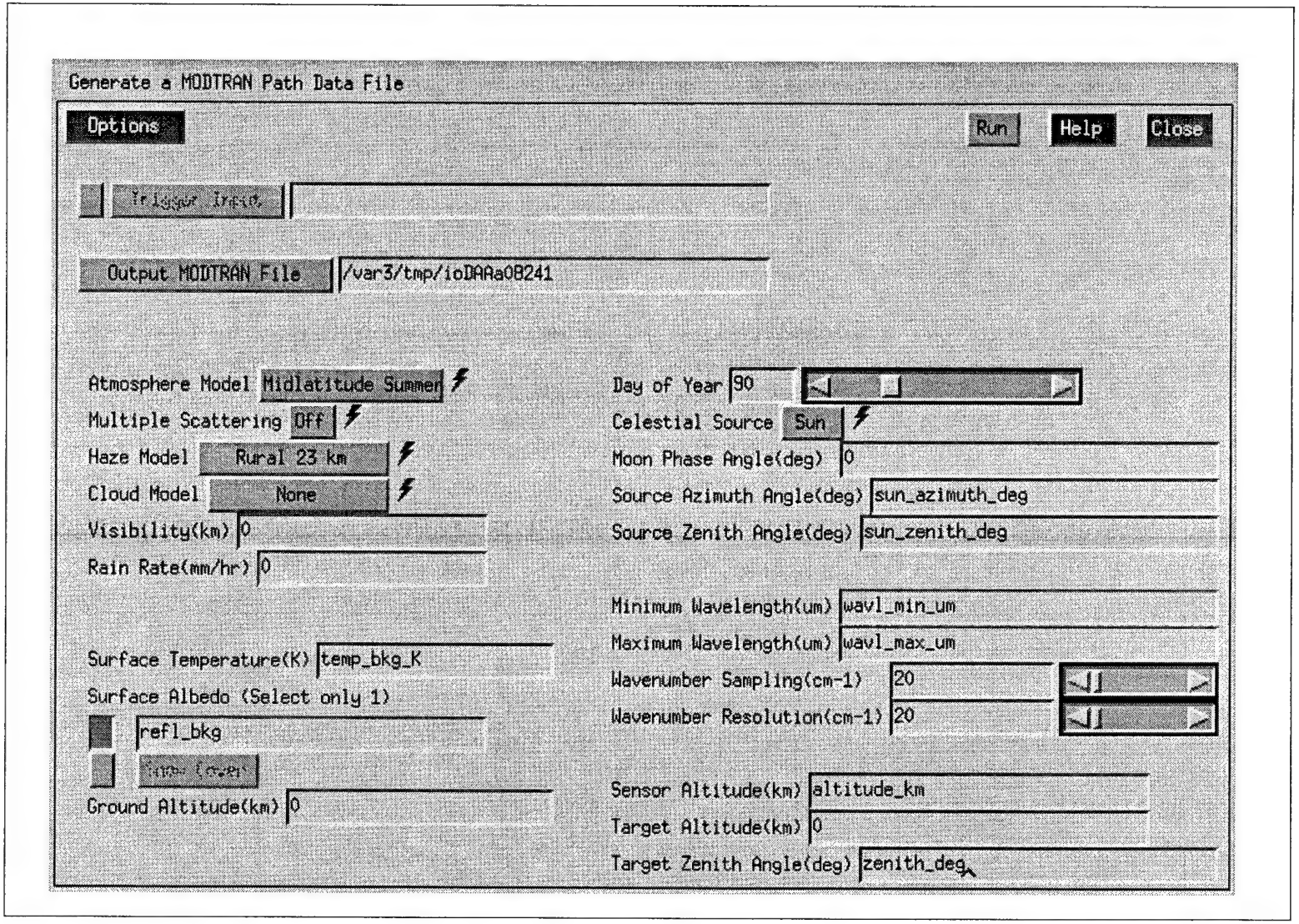

Figure 5-3: MODTRAN Input Form

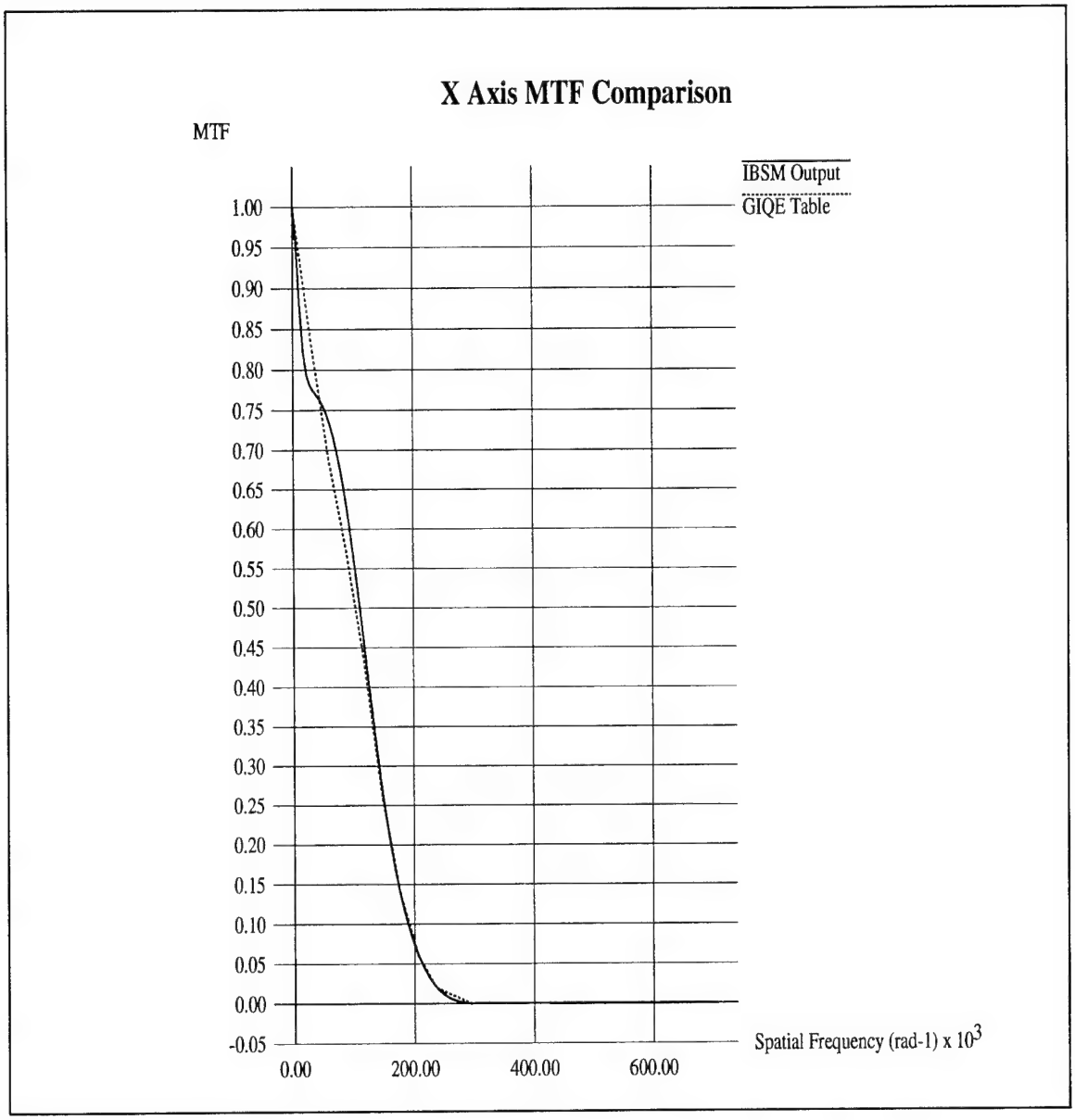

Figure 5-4: Comparison of Test Case MTFs (X-axis)

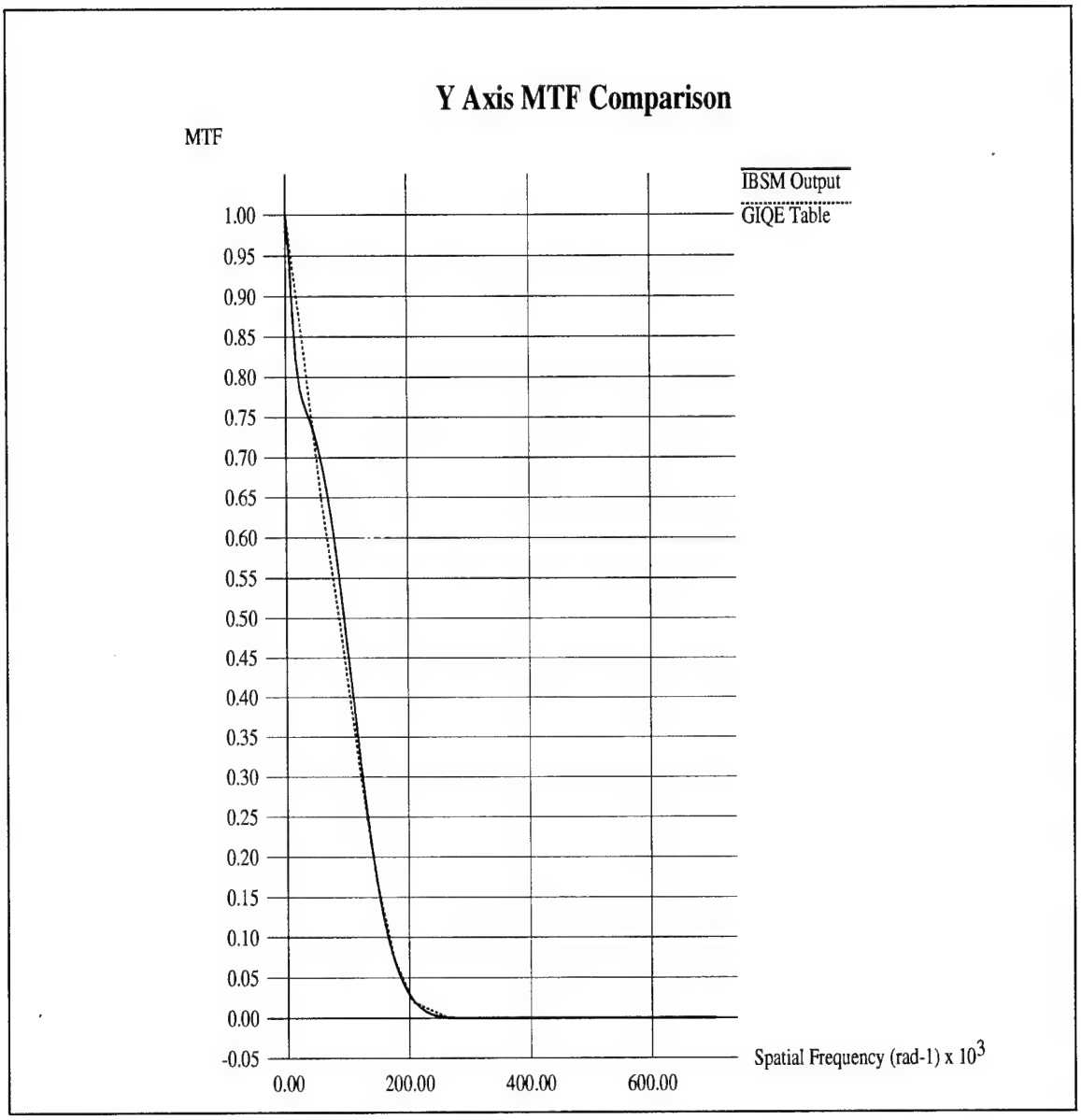

Figure 5-5: Comparison of Test Case MTFs (Y-axis)

Table 5-2: GIQE Test Metric Comparison

	GIQE Documentation	IBSM Example
GSD	3.8 inches	3.79 inches
RER	0.57	0.582
H	0.94	0.938
G	2.8	2.8
Signal	24,000 electrons	15,700 electrons
Noise	312 rms electrons	235 rms electrons
SNR	77	66.6
NIIRS	7.6	7.75

5.2 MTF ANALYSIS

The second example is an illustration of the individual MTF components for the GIQE test case. The workspace is depicted in Figure 5-6. In this case, there are parallel paths for generating the MTF functions (in the X direction) due to the atmospheric path, optics, line-of-sight motion, and detector. Figure 5-7 provides the resulting MTF plots.

5.3 PARAMETRIC PERFORMANCE LOOP

The third example is an illustration of computing sensor performance (SNR and NIIRS) as a function of an independent variable, specifically ground range. Essentially, the model from Section 5.1 is inserted inside a loop which adjusts ground range from 0 to 100 km in 5 km increments. The workspace is given in Figure 5-8. In addition, a conditional branch exists which sets an update parameter for the *Sensor performance* modules such that they initialize the output data files on the first iteration and append them on subsequent iterations. The resulting SNR and NIIRS plots versus ground range are given as Figures 5-9 and 5-10. These differ slightly from the test case because MTF compensation was not included in the loop.

5.4 IMAGE BASED ANALYSIS

The fourth example represents the primary intended use of the IBSM capabilities. Using the same hypothetical sensor description, the workspace depicted in Figure 5-11 simulates an output image from a high resolution, high SNR input scene, and extracts edge response functions from a test border around the image to compute NIIRS.

The input scene is appropriately scaled spatially and in (apparent) reflectivity units. The model proceeds as follows. First, an exposure control sequence is executed to avoid detector saturation. Next, the input scene is converted into a line-of-sight radiance image and propagated through the atmospheric path. The image is then transformed to the spatial spectrum domain and passed through a series of OTFs. After transforming back to the image domain, the radiance image is converted to a focal plane irradiance image, then a detected photoelectron image, sampled at the detector pitch, perturbed with random noise, and quantized. Finally, the edge response functions are extracted and sensor performance is computed.

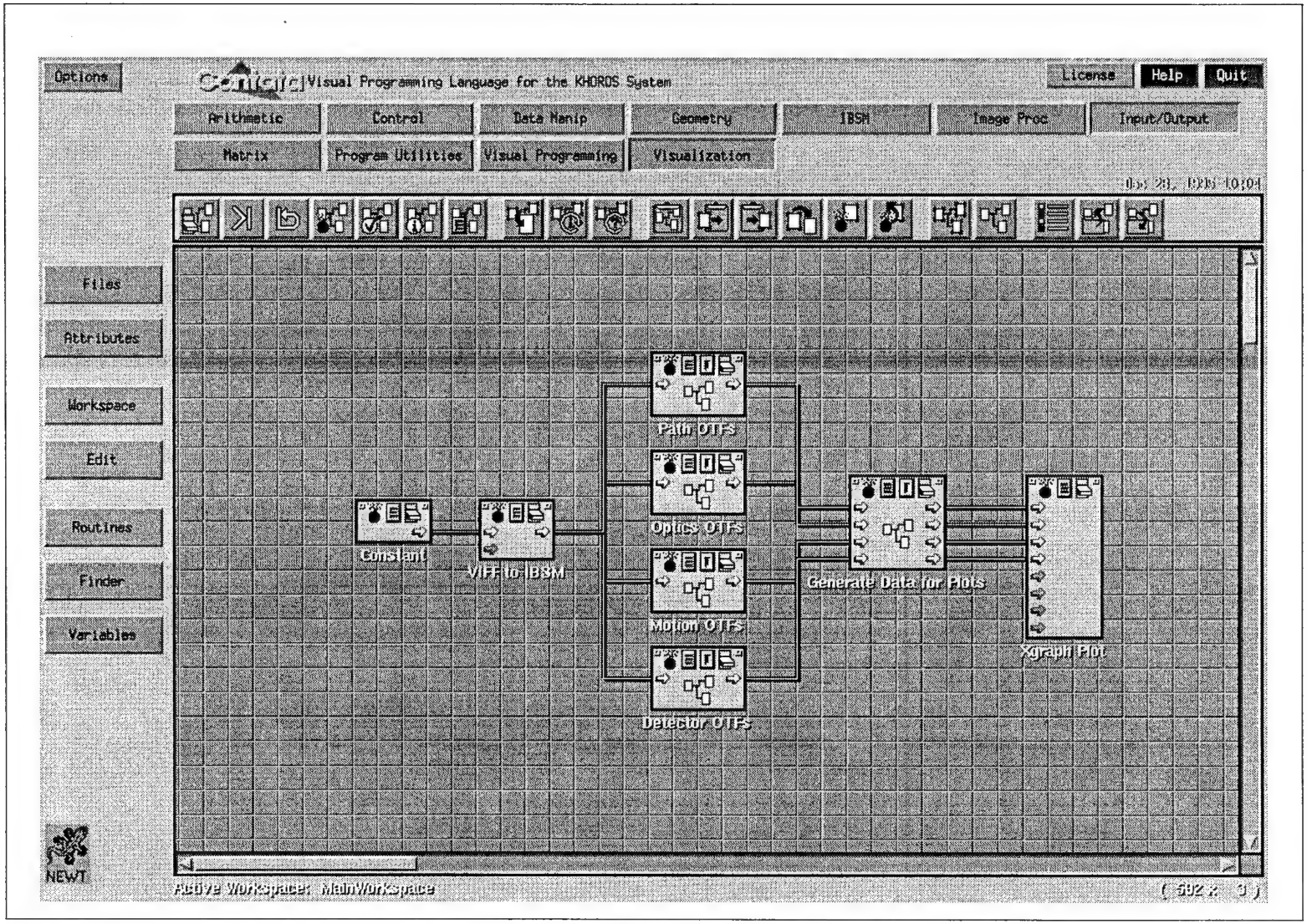

Figure 5-6: Workspace Corresponding to MTF Analysis

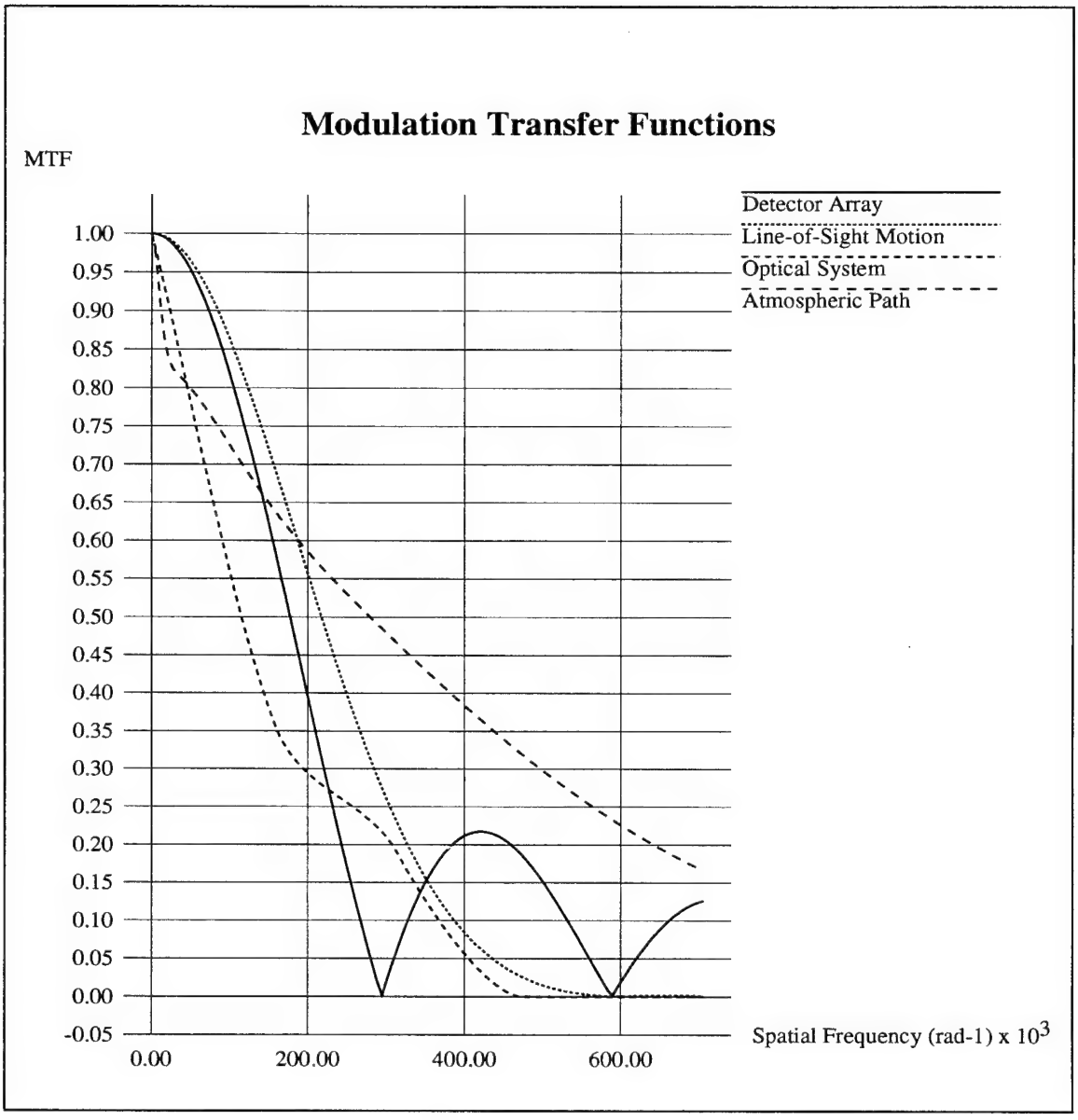

Figure 5-7: MTF Analysis Results (X-axis)

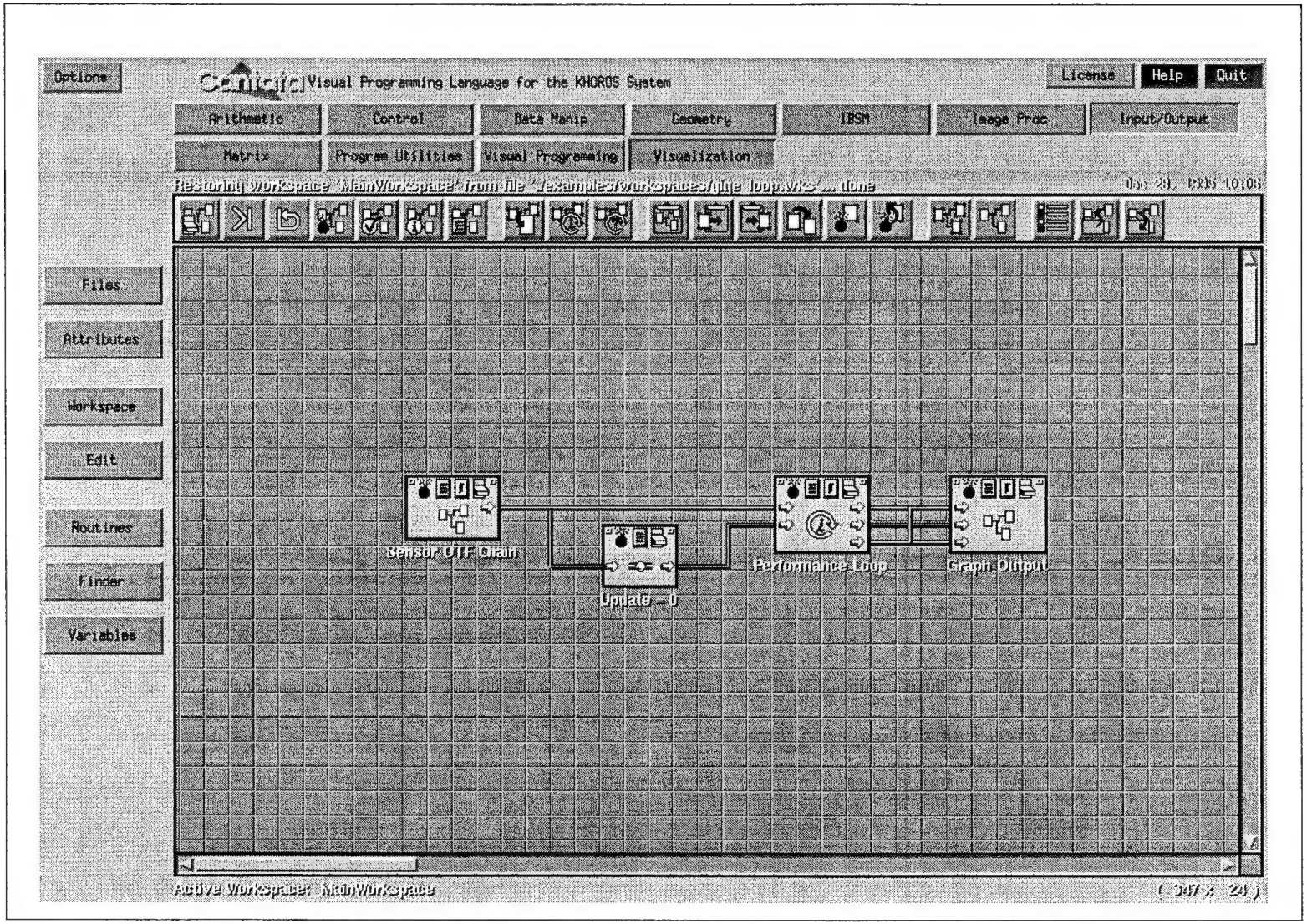

Figure 5-8: Workspace Corresponding to Parametric Performance Loop

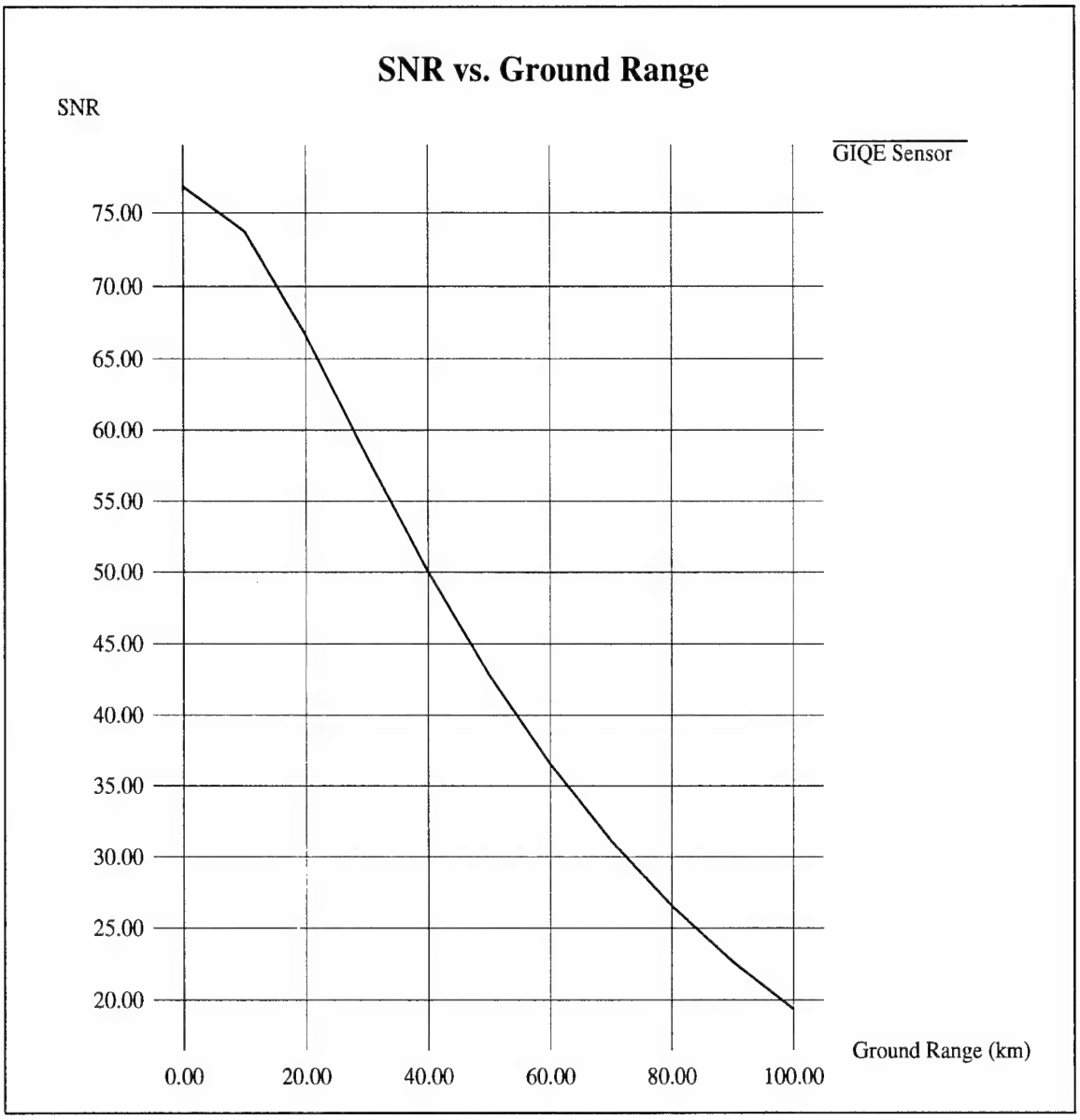

Figure 5-9: Signal-to-Noise Ratio Versus Ground Range

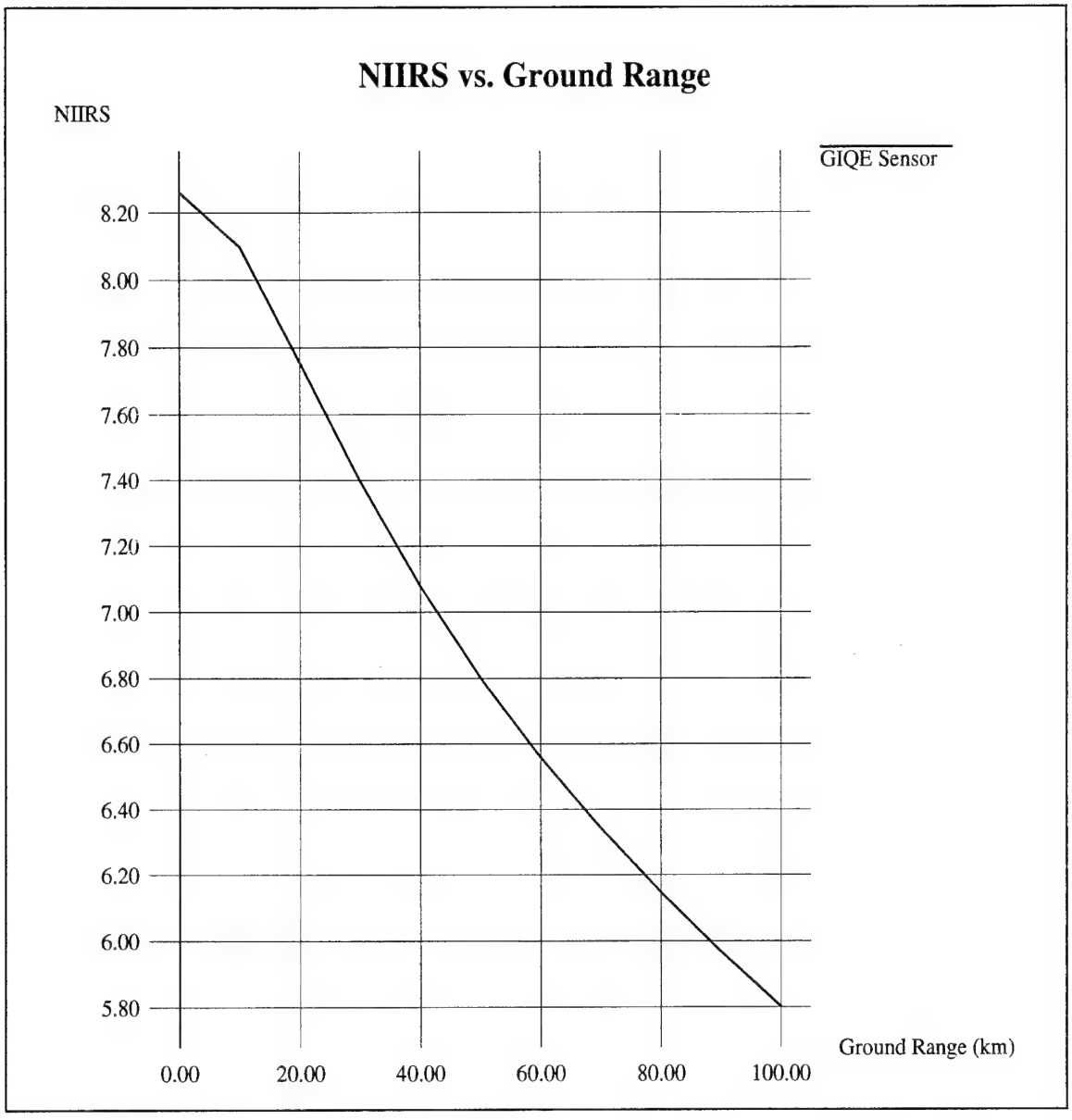

Figure 5-10: Image Quality Versus Ground Range

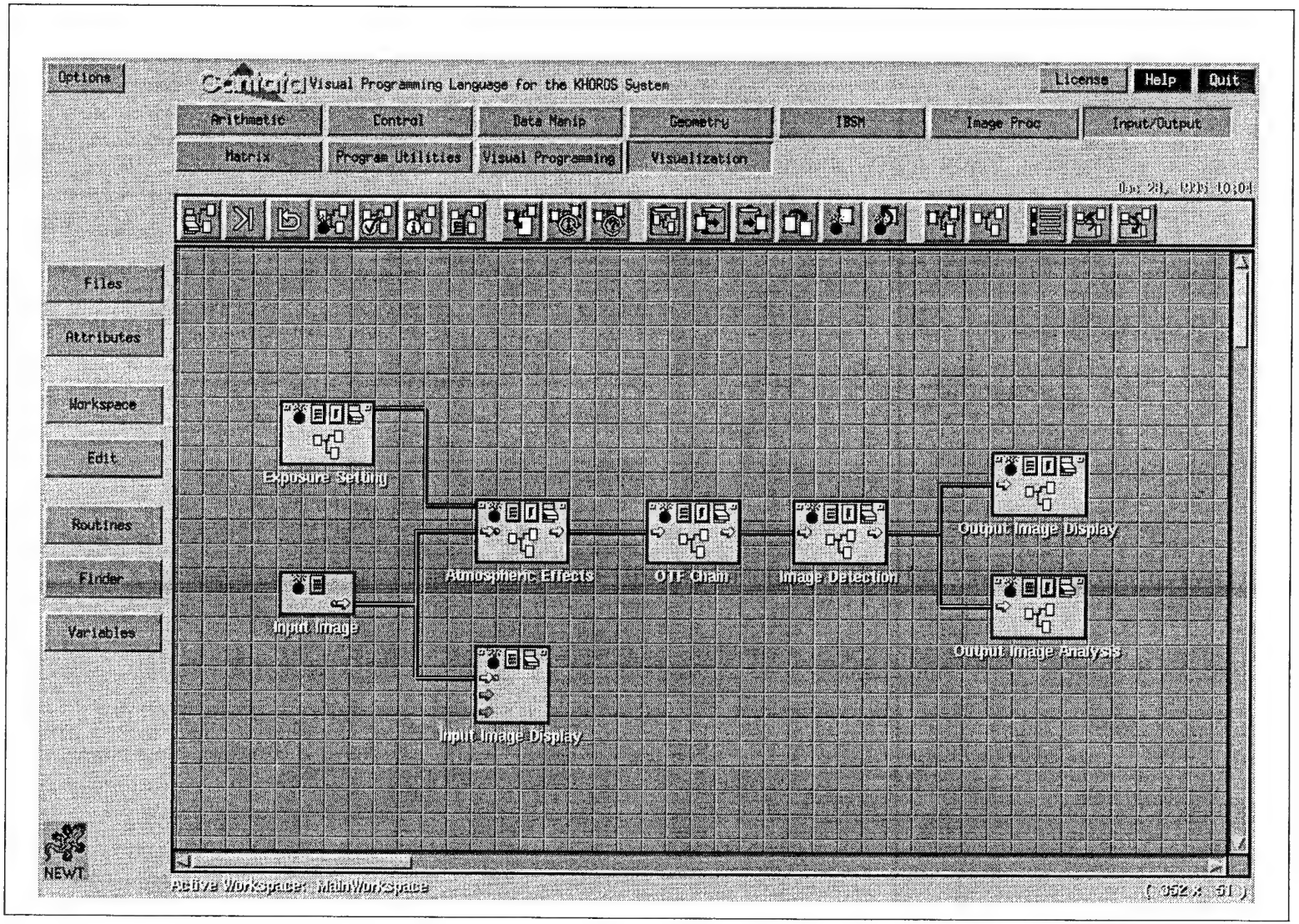

Figure 5-11: Workspace for Image Based Analysis

The input scene is shown in Figure 5-12, and can be compared to the simulated output image shown in Figure 5-13. The edge response functions are illustrated in Figure 5-14, and the sensor performance computations are summarized in Table 5-3.

Utilizing an appropriate input scene perspective and adjusting the ground range to 100 km, an analogous simulation with a greater amount of image degradation is executed. The results are shown in Figures 5-15 to 5-17 and Table 5-4.

5.5 DATA LINK MODELING

The final example is an illustration of utilization of the data link toolbox which has been added to IBSM. Figure 5-18 provides the workspace for modeling the data link summarized in Table 5-5, consisting of JPEG compression/decompression, a finite buffer, and channel errors. The input scene is shown as Figure 5-19. This image exhibits a fairly nonuniform amount of compression over the image since the image is much more detailed near the center. The bits/block file, shown as Figure 5-20, illustrates this fact. For the chosen parameters and image, the buffer overflows near the end of the image (4096th block), as shown in the buffer occupancy plot in Figure 5-21. The output image (Figure 5–22) illustrates terminal failure of the decompressor at this point due to the lost information, as well as temporary failure for channel bit errors (recovery at the restart codes).

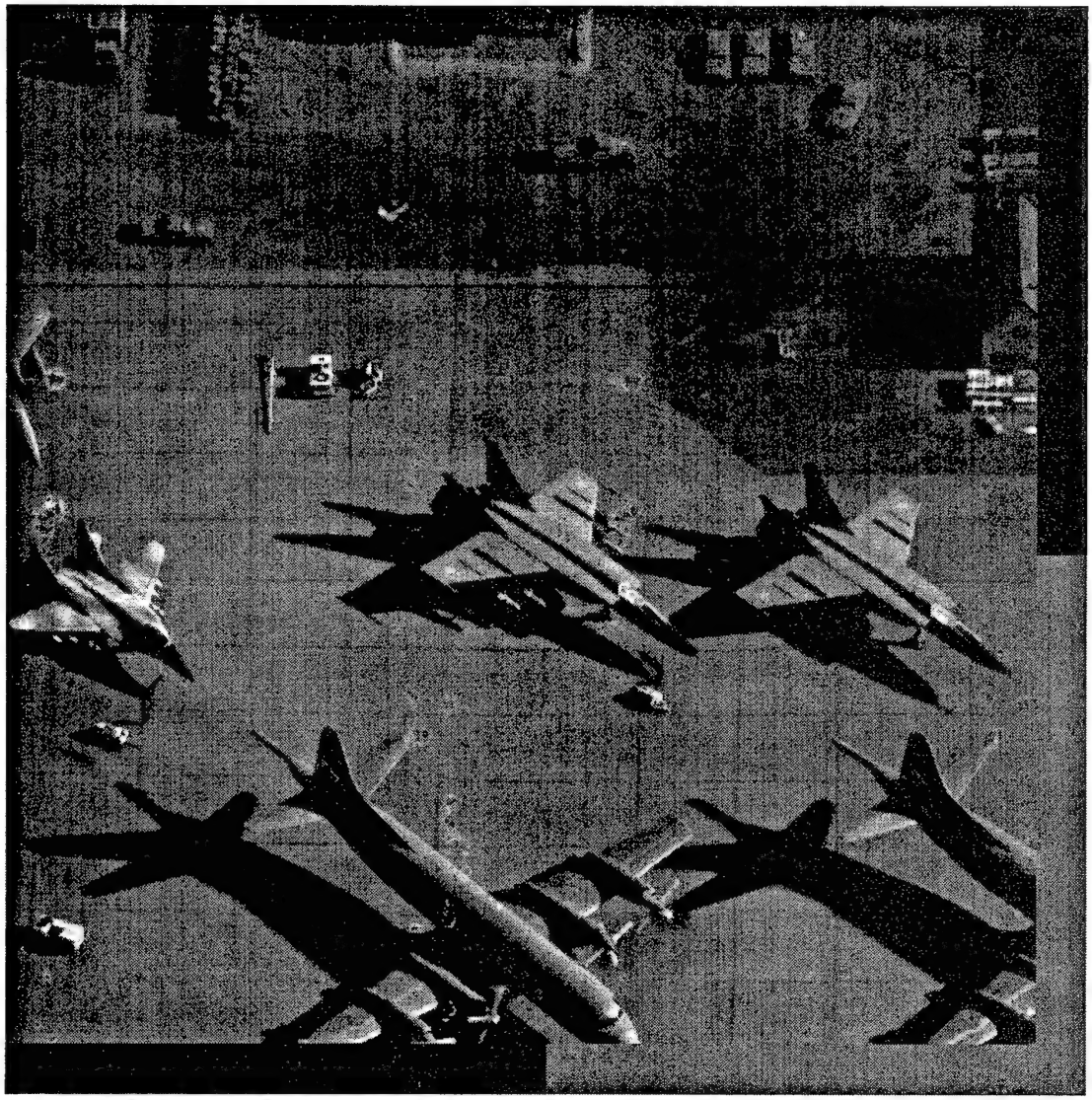

Figure 5-12: Input Image for 20 km Ground Range Simulation

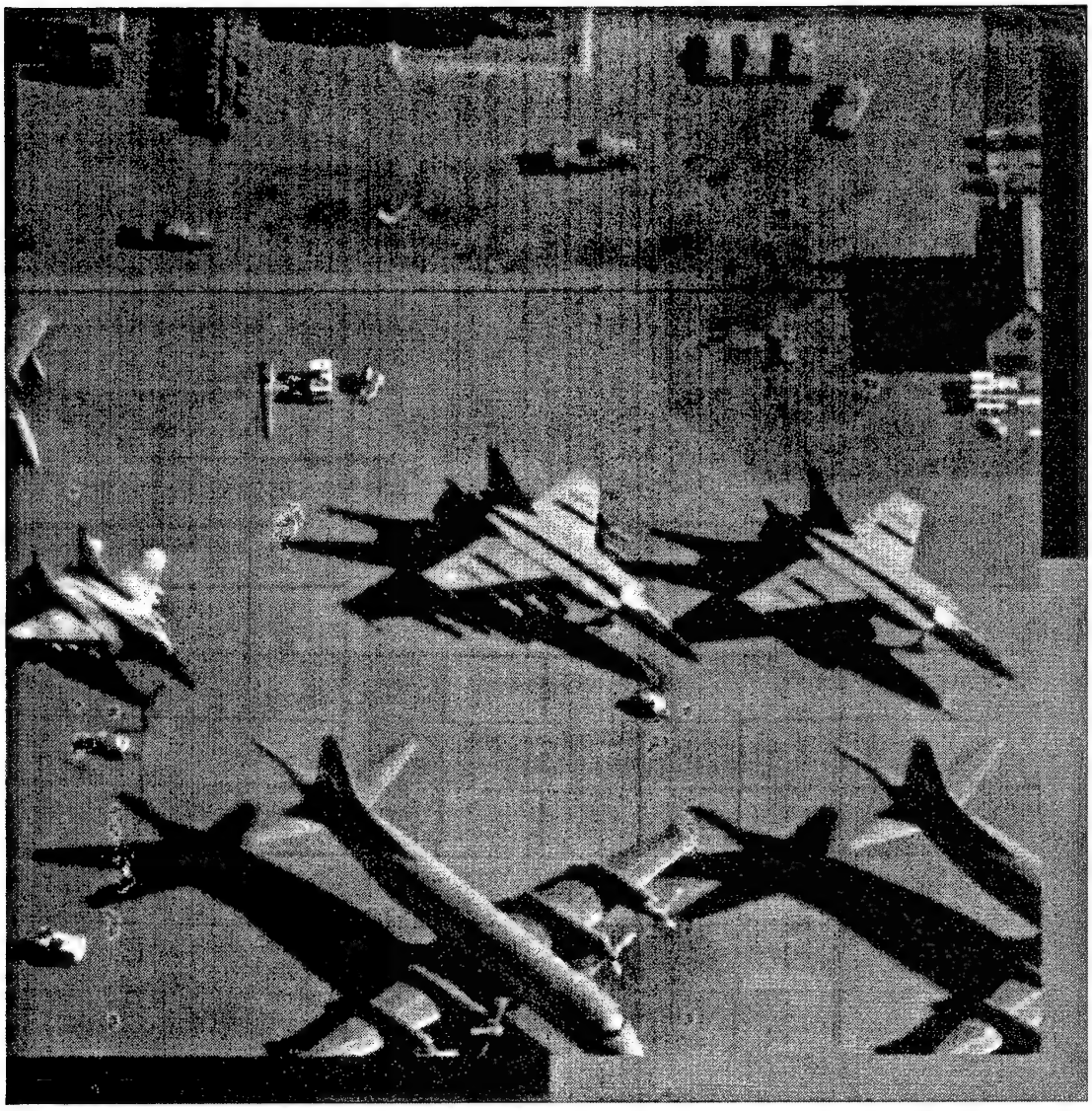

Figure 5-13: Output Image for 20 km Ground Range Simulation

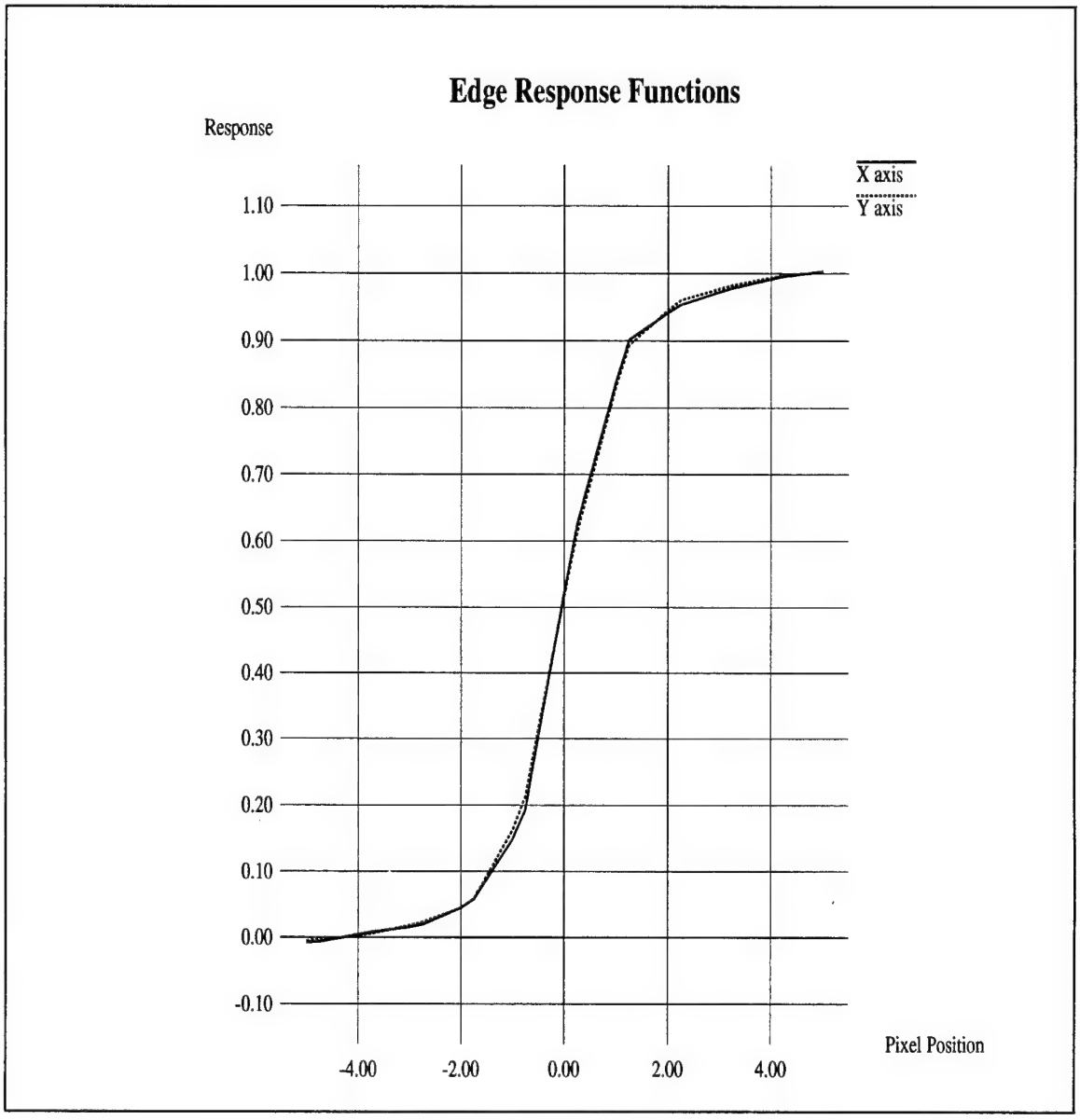

Figure 5-14: Edge Response Functions (20 km)

Table 5-3: Sensor Performance at 20 km Ground Range

Relative Edge Response
RERx = 0.394
RERy = 0.371
RER = 0.383

Edge Height Overshoot
Hx = 0.901
Hy = 0.893
H = 0.897

Ground Sample Distance
GSDx = 3.793 inches
GSDy = 3.793 inches
GSD = 3.793 inches

Signal to Noise Ratio
Tgt mean = 73159.12
Tgt stdev = 290.65
Bkg mean = 44882.627
Bkg stdev = 242.39
SNR = 97.29

NIIRS Rating = 7.25

Figure 5-15: Input Image for 100 km Ground Range Simulation

Figure 5-16: Output Image for 100 km Ground Range Simulation

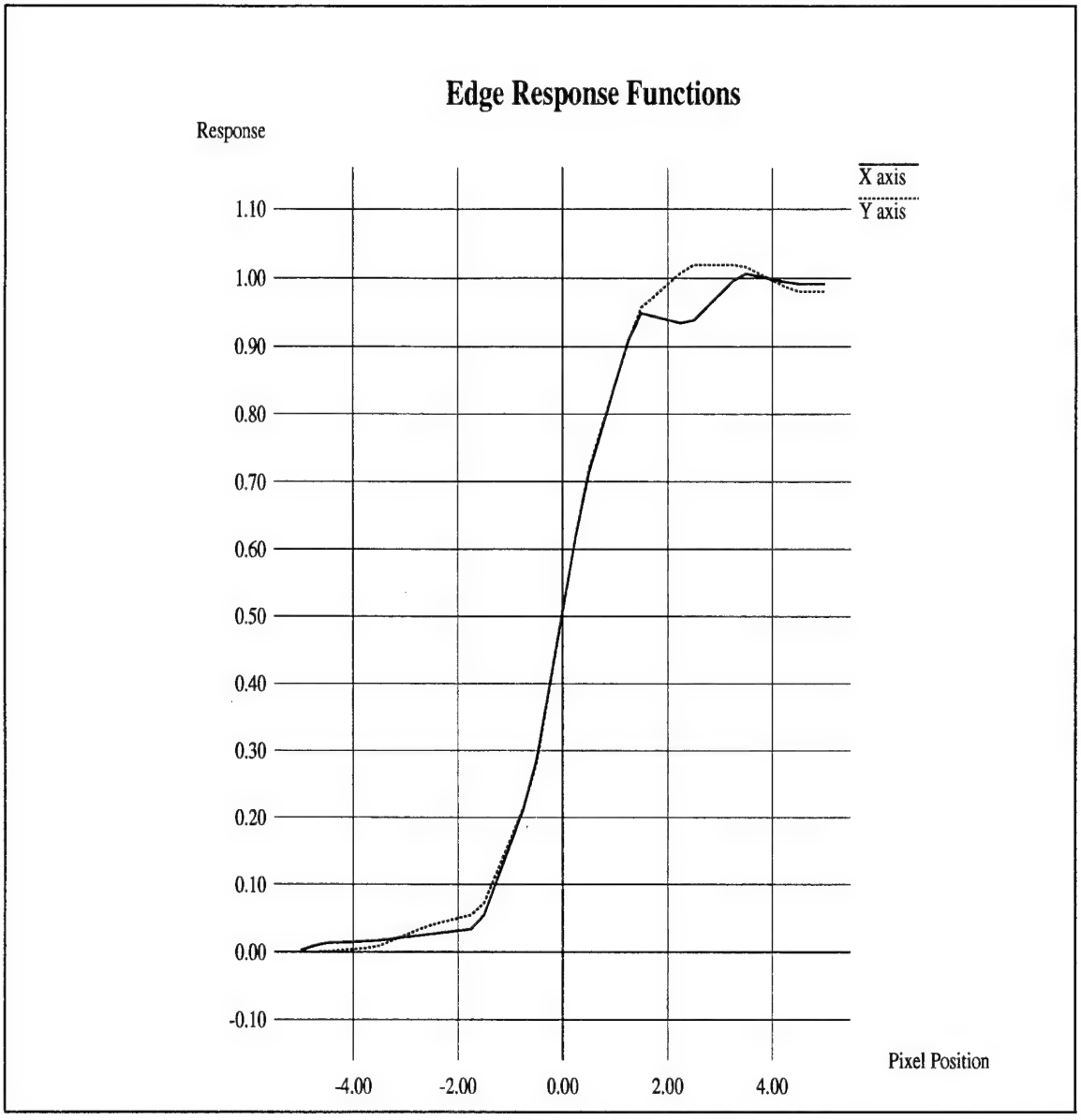

Figure 5-17: Edge Response Functions (100 km)

Table 5-4: Sensor Performance at 100 km Ground Range

Relative Edge Response
RERx = 0.427
RERy = 0.437
RER = 0.432

Edge Height Overshoot
Hx = 0.977
Hy = 1.019
H = 0.998

Ground Sample Distance
GSDx = 13.741 inches
GSDy = 13.741 inches
GSD = 13.741 inches

Signal to Noise Ratio
Tgt mean = 96869.36
Tgt stdev = 352.00
Bkg mean = 89230.78
Bkg stdev = 391.26
SNR = 21.70

NIIRS Rating = 5.38

Table 5-5: Data Link Modeling Parameters

Image Compression	
Type:	JPEG (12 bit)
Image Quality Factor:	3
Blocks per Restart Code:	8
Buffer	
Size	64 KByte
Input Rate	1 MWord/sec
Output Rate	2 Mbits/sec
Delay to Empty	6.4 msec
Channel Errors	
Flip Errors	10 errors/sec
Insert Errors	None
Delete Errors	None
Burst Errors	5 errors/sec
Burst Length	1000 bits (variable)

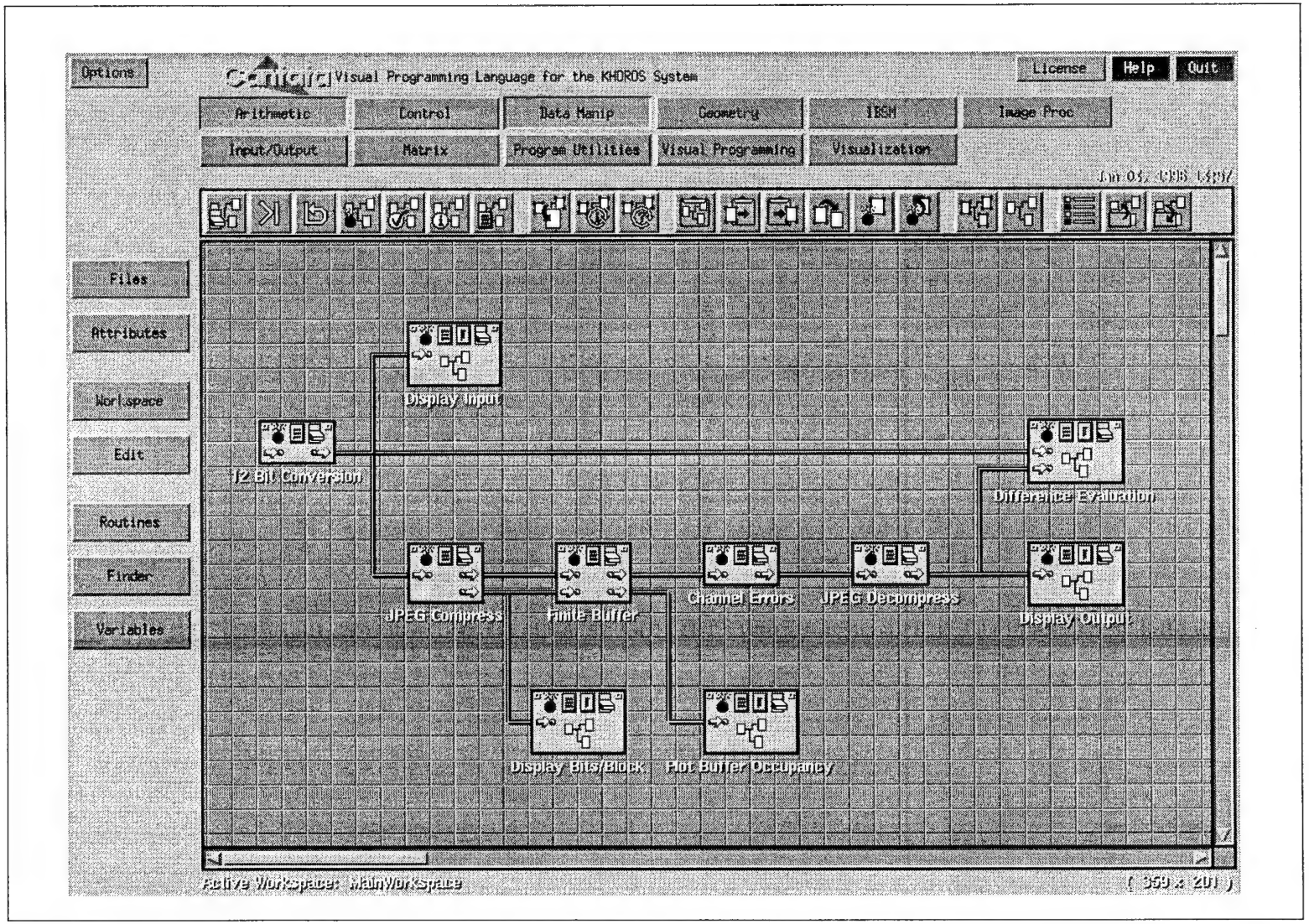

Figure 5-18: Workspace Corresponding to Data Link Effects Simulation

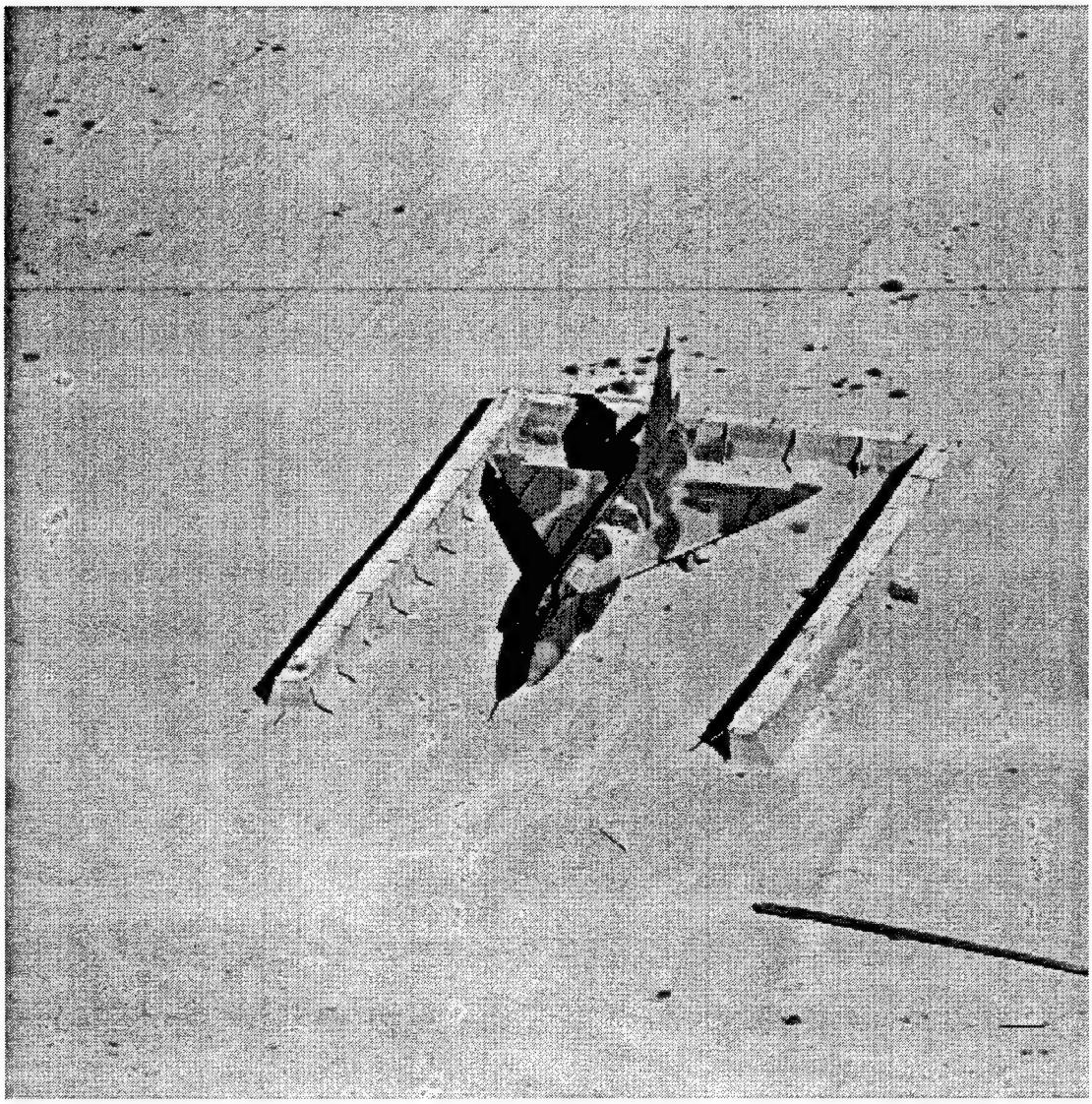

Figure 5-19: Input Image for Data Link Effects Simulation

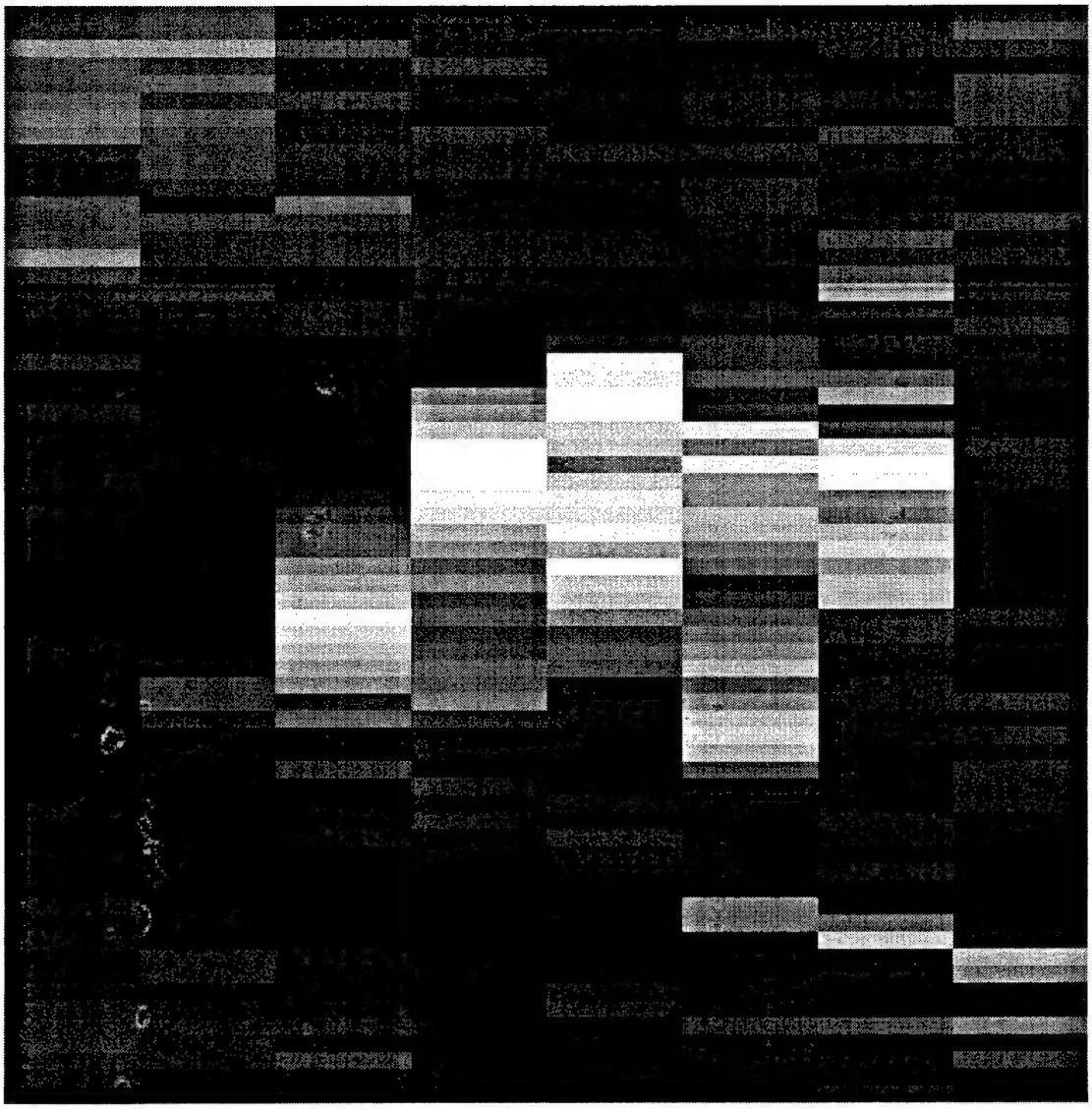

Figure 5-20: Bits/Block Image for JPEG Compressed Image

6.0 SOFTWARE OVERVIEW

As depicted in Figure 6-1, the IBSM software consists of primarily three functional layers of software: a Khoros executive layer containing file I/O and graphical user interface, a Khoros library function layer containing functional interface between executive layer and the analytical code, and the sensor modeling library containing the analytical modeling code. Numerical Recipes and MODTRAN are used for analytical computations and atmospheric modeling.

6.1 KHOROS EXECUTIVE LAYER

Table 6-1 summarizes the routines contained in the Khoros executive layer. Each routine is a UNIX command line executable function, but also contains the appropriate code to run within Cantata. This additional code includes the graphical user interface and on-line documentation. The appendix provides the on-line documentation (Man Pages) for all toolbox functions.

6.2 KHOROS LIBRARY FUNCTION LAYER

The routines contained in the Khoros library function layer are associated with the executive layer functions in Table 6-1. These routines are C functions which are called by Khoros executive functions. The majority of these functions contain the software which loops over the multiple input image bands and calls specific analytical function to appropriately process each image band. In the case of the parametric functions, these functions contain module-specific analytical code as well.

6.3 SENSOR MODELING LIBRARY

Table 6-2 summarizes the routines contained in the sensor modeling library. These routines are C functions which form the analytical modeling basis of the IBSM software. These routines are called by the Khoros library functions to perform the majority of the physical modeling described in Section 3.

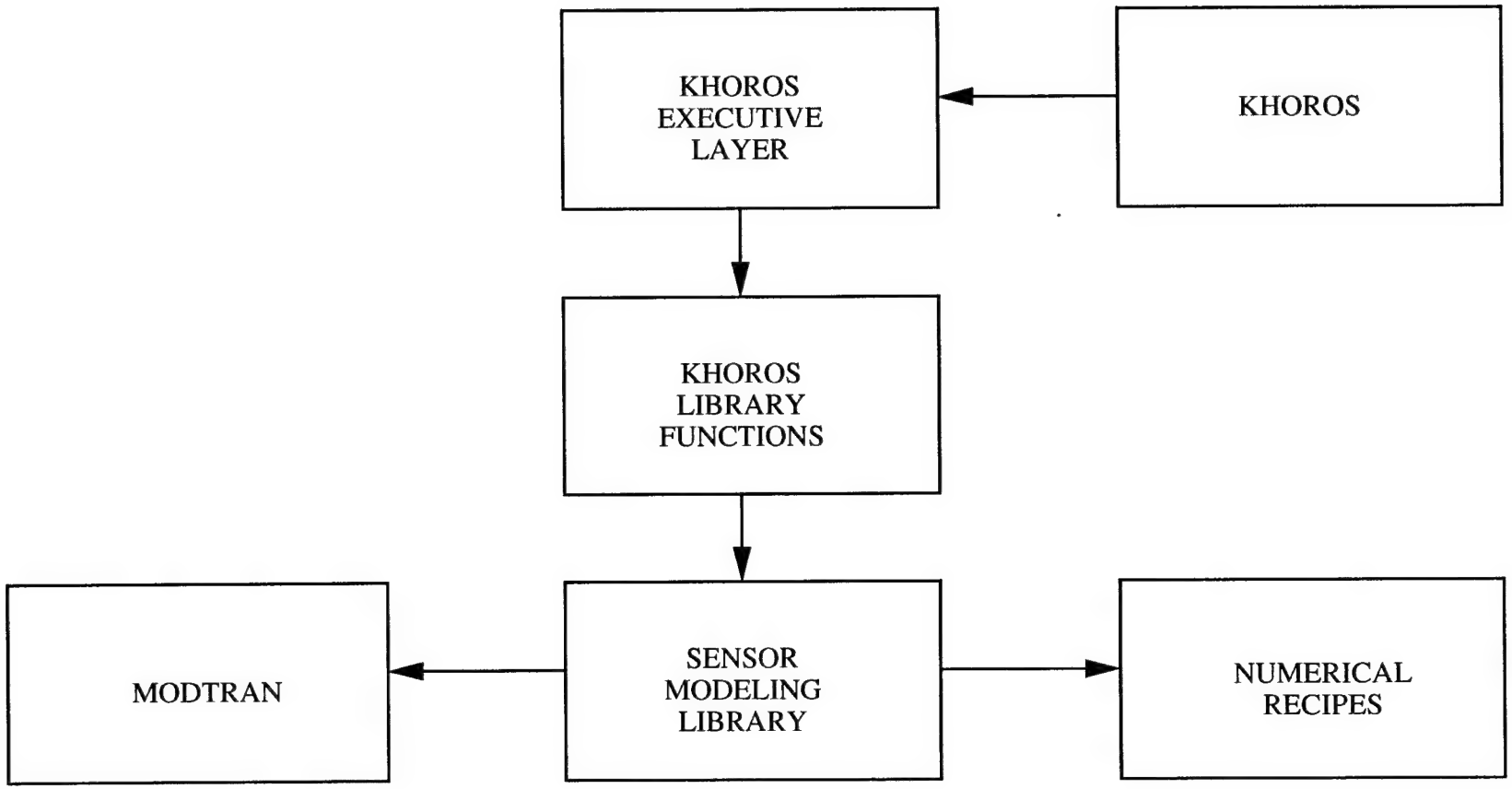

Figure 6-1: IBSM Software Structure

Table 6-1: Khoros Executive Layer Routines

Toolbox	Function Name	Executive Routine	Library Function Routine
Atmosphere	Modtran Input	atmodtran	
	Path Trans/Rad	atpatrra	katmos_path
	Turbulence OTF	atturbotf	kotf_turb
	Aero-Optic OTF	ataerotf	kotf_aero
Optics	Diffraction OTF	opdiffotf	kotf_diffr
	Defocus OTF	opdefotf	kotf_blur
	Jitter OTF	opjittotf	kotf_jitter
	Drift OTF	opdrifotf	kotf_drift
	Wavefront OTF	opwaveotf	kotf_wave
	User OTF (1-D)	opuserotf	kotf_user
	User OTF (2-D)	opusrotf2	kotf_user2d
	Radiometry	opradiom	kradiometry
Detector	Detector Size OTF	desizeotf	kotf_detsize
	TDI OTF	detdiotf	kotf_tdi
	CTE OTF	decteotf	kotf_cte
	Diffusion OTF	dediffotf	kotf_detdiff
	Detection	dedetect	kdetection
	Noise	denoise	knoise
	Quantization	dequantiz	kquantize
	Sampling	desamplin	ksampling
	Nonuniformity	denonunif	knonunit
Data Link	JPEG Compress	dljpegcomp	jpeg_compress
	JPEG Decompress	dljpegdecomp	jpeg_decomp
	TSVQ Compress/Decompress	dltsvq	tsvq_(de)compress
	Wavelet Comp/Decomp	dlwavelet	wave
	Finite Buffer	dlbuffer	buffer
	Channel Errors	dlchannel	channel
Performance	Create Spectra	pecrespec	kspectra
	Create MTF	pecremtf	kmtf
	Sensor Performance	pesensper	kperf
	Exposure Control	peexpctl	kexp_cntrl
	Image Quality	peimgqual	kqimage
Processing	Image Interlace	printerlace	
	Aggregation	praggreg	
Utilities	VIFF to IBSM	viff2ibsm	
	List IBSM File Data	ibsminfo	
	Transform	utxform	ktransform
	Create Plot Data	utcreplot	
	Create Radiance Image	utcreradi	kradimg
	Ground/Angle Transform	utgrdangl	kgndang
	Image Interlace	utinterlace	
	Load Variables	utloadvars	
	Xgraph Plot	utxgraph	

Table 6-2: Sensor Modeling Library Routines

Group	Name	Function
Atmosphere	atmos_path	Apply atmospheric path transmission and radiance to image
	otf_turb	Generate and apply path turbulence OTF to spatial spectrum
	otf_aero	Generate and apply aero-optic boundary layer OTF to spatial spectrum
	atmos_r0	Compute atmospheric path correlation diameter
	atmos_cndata	Compute index structure profile along path
	modtran_path	Compute atmospheric transmission and path radiance
	modtran_rad	Compute solar or lunar direction and diffuse radiance
Optics	otf_diffr	Generate and apply aperture diffraction OTF to spatial spectrum
	otf_blur	Generate and apply gaussian blur OTF to spatial spectrum
	otf_jitter	Generate and apply line-of-sight jitter OTF to spatial spectrum
	otf_drift	Generate and apply line-of-sight drift OTF to spatial spectrum
	otf_wave	Generate and apply wavefront error OTF to spatial spectrum
	otf_user	Generate and apply 1-D user defined OTF to spatial spectrum
	otf_user2d	Generate and apply 2-D user defined OTF to spatial spectrum
	radiometry	Perform optical system radiometric operations
Detector	detection	Perform detector photoelectronic conversion
	gauss_noise	Apply gaussian distributed noise to image
	shot_noise	Apply signal dependent, Poisson distributed noise to image
	otf_detsize	Generate and apply detector footprint OTF to spatial spectrum
	otf_detdiff	Generate and apply detector diffusion OTF to spatial spectrum
	otf_tdi	Generate and apply time-delay-integrate OTF to spatial spectrum
	otf_cte	Generate and apply charge transfer inefficiency to spatial spectrum
	sample	Sample image based on detector pitch
	quantize	Quantize image based on scalar truncation
	nonunif	Apply effects of gain and/or offset nonuniformity
Utilities	transform	Convert between image and spatial spectrum domains
Miscellaneous	copyhdr	Copy header structure
	blackbody	Compute blackbody spectral radiance
	linint	Perform linear interpolation
	rect	Compute rect function
	sinc	Compute sinc function
	tri	Compute tri function

7.0 REFERENCES

1. M.T. Eismann and S.D. Ingle, *Utility Analysis of High Resolution Multispectral Imagery*, Volume 3: Image Based Sensor Model (IBSM) Users Manual, Final Report to ASC/REFQ, Contract Number DLA900-88-D-0392, Delivery Order 57, Environmental Research Institute of Michigan, May 1995.

2. F.X. Kneizys, et al., *Users Guide to LOWTRAN 7*, AFGL-TR-88-D177, Air Force Geophysics Laboratory, Hanscom AFB, MA, August 1988.

3. A. Berk, et al., *MODTRAN: A Moderate Resolution Model for LOWTRAN 7*, GL-TR-89-0122, Geophysics Laboratory, Hanscom AFB, MA, April 1989.

4. J.W. Goodman, *Statistical Optics*, (Wiley, 1985), p. 439.

5. D.L. Fried, "Optical Resolution Through a Randomly Inhomogeneous Medium for Very Long and Very Short Exposures,"*J. Opt. Soc. Am.*, vol. 56 (October, 1966), p. 1372.

6. D. L. Fried, "Limiting Resolution Looking Down Through the Atmosphere,"*J. Opt. Soc. Am.*, vol. 56 (October 1986) p. 1380.

7. R.R. Beland, "Propagation Through Atmospheric Optical Turbulence," in *The Infrared and Electro-Optical Systems Handbook,* Volume 2: Atmospheric Propagation of Radiation (SPIE, 1993), pp. 157-232.

8. F. Roddier, "The Effects of Atmospheric Turbulence in Optical Astronomy," in Progress in Optics (North-Holland, 1981), Ch. 5.

9. R.E. Hufnagel, "Propagation Through Atmospheric Turbulence," in *The Infrared Handbook* (Office of Naval Research, 1978), Ch. 6.

10. A.E. Guryanov, "Astronomical Image Quality and the Vertical Distribution of Turbulence Optical Interference in the Night Atmosphere," *Sov. Astron*n., vol. 28 (June, 1984), p. 343.

11. K.G. Gilbert, et al., "Aerodynamic Effects," in *The Infrared and Electro-Optical Systems Handbook*, Volume 2: Atmospheric Propagation of Radiation (SPIE, 1993), pp. 235-246.

12. J.E. Craig, *Aero-Optics Calculations for a Sphere-Cylinder Body*, Physical Research Report to ERIM (November, 1991).

13. J.W. Goodman, *Introduction to Fourier Optics* (McGraw-Hill, 1968), pp. 112-120.

14. E.L. O'Neill, "Transfer Function for an Annular Aperture," *J. Opt. Soc. Am.*, vol. 46 (April, 1956), p. 285.

15. J.D. Howe, "Electro-Optical Imaging System Performance Prediction," in *The Infrared and Electro-Optical Systems Handbook*, Volume 4: Electro-Optical Systems Design, Analysis, and Testing, (SPIE, 1993) pp. 69-70.

16. Ibid, p. 69.

17. J.W. Goodman, *Statistical Optics* (Wiley, 1985), p. 376-81.

18. Howe, p. 70.

19. Howe, p. 103-4.

20. D.H. Seib, "Carrier Diffusion Degradation of Modulation Transfer Function in Charge Coupled Imagers," *IEEE Trans. Electron Devices*, vol. ED-21 (March, 1974) p. 210.

21. W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard* (Van Nostrand Reinhold, 1993).

22. *Joint Photographic Experts Group (JPEG) Image Compression for the National Imagery Transmission Format Standards (NITFS)*, MIL-STD-188-198A.

23. Optivision Co., 3450 Hillview Dr., Palo Alto, CA 94304.

24. A Gersho and R.M. Gray, *Vector Quantization and Signal Compression* (Kluwer Academic Publishers, 1992).

25. J. Goldschneider, "TSVQ Programs," ftp://isdl.ee.washington.edu/pub/VQ/code/tsvq/, February 1992 to March 1994.

26. X. Wang, *et al*, "Wavelet-Based Image Coding using Nonlinear Interpolative Vector Quantization," *IEEE Transactions on Image Processing*, vol. 5 (March 1996), pp. 518-522.

27. P. Westernick, *et al*, "Subband Coding of Images Using Vector Quantization," *IEEE Transactions on Communications*, vol. 36 (June 1988), pp. 713-719.

28. S. Ross, *Introduction to Probability Models^(Academic Press, 1989).*

29. *General Image Quality Equation (GIQE) Users Guide Version 3.0*, Report to HAE UAV Tier II+ Distribution (December, 1994).

30. W.H. Press, et al., *Numerical Recipes in C* (Cambridge Press, 1992) Ch. 12.

31. *Khoros Visual Programming Manual*, Chapter 1: Cantata (Khoral Research, 1993).

Appendix

Man Pages

Appendix - Man Pages

A. Atmosphere Man Pages

A.1. ataerotf — *OTF due to boundary layer aero-optic turbulence*

Object Information

Category:	*IBSM*	Subcategory:	*Atmosphere*
Operator:	*Aero-Optic OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an aero-optic OTF to a spatial spectrum for a specified turbulent boundary layer. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for ataerotf: OTF due to boundary layer aero-optic turbulence
% ataerotf
  -i     (infile)  Name of input IBSM spatial spectrum file
  -o     (outfile) Name of output IBSM spatial spectrum file
  -vel   (float)   Sensor platform velocity in m/s
          vel >= 0.0
  -rho   (float)   Air density in kg/m3 at sensor altitude
          rho >= 0.0
  -tbl   (float)   Boundary layer thickness in m at sensor aperture
          tbl >= 0.0
```

A.2. atmodtran — *Generate a MODTRAN Path Data File*

Object Information

Category:	*IBSM*	Subcategory:	*Atmosphere*
Operator:	*MODTRAN Input*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function provides a user interface for generating a MODTRAN path data file as an input to IBSM functions that execute MODTRAN, including "Path Trans/Rad", "Create Spectra", and "Create Radiance Image". The output file is in ASCII format.

Command Line Arguments

```
Usage for atmodtran: Generate a MODTRAN Path Data File
% atmodtran
  -o         (outfile) Name of output ASCII MODTRAN path data file
  -model     (cycle)   Model atmosphere
           0  "Tropical",
           1  "Midlatitude Summer",
           2  "Midlatitude Winter",
           3  "Subartic Summer",
           4  "Subartic Winter",
           5  "1976 US Standard"

  -iday      (integer) Day of year in days starting Jan. 1
              1 <= iday <= 365
  -imult     (cycle)   Multiple scattering switch
           0  "Off",
           1  "On"

  -isourc    (cycle)   Celestial source
           0  "Sun",
           1  "Moon"

  -ihaze     (cycle)   Aerosol haze model
           0  "None",
           1  "Rural 23 km",
           2  "Rural 5 km",
           3  "Navy Maritime",
           4  "Maritime 23 km",
           5  "Urban 5 km",
           6  "Troposheric 50 km",
           7  "Fog1 0.2 km",
           8  "Fog2 0.5 km",
           9  "Desert"

  -anglem    (float)   Moon phase angle in degrees
              unbounded
  -icld      (cycle)   Cloud model
           0  "None",
           1  "Cumulus",
           2  "Altostratus",
           3  "Stratus",
           4  "Stratocumulus",
           5  "Nimbostratus",
           6  "Drizzle",
           7  "Light Rain",
           8  "Moderate Rain",
           9  "Heavy Rain",
          10  "Extreme Rain",
          11  "Standard Cirrus",
          12  "Subvisual Cirrus",
          13  "NOAA Cirrus"
```

```
-parm1     (float)   Source azimuth angle in degrees (behind sensor = 180)
           unbounded
-vis       (float)   Visibility in km (0.0 =  default for haze model)
           vis >= 0.0
-parm2     (float)   Source zenith angle in degrees (vertical = 0)
           unbounded
-rainrt    (float)   Rain rate in mm/hr
           rainrt >= 0.0
-wavl1     (float)   Minimum wavelength in microns
           unbounded
-wavl2     (float)   Maximum wavelength in microns
           unbounded
-tbound    (float)   Surface temperature in K
           tbound >= 0.0
-idv       (integer) Spectral sampling in cm-1
           1 <= idv <= 2.14748e+09
-ifwhm     (integer) Spectral resolution in cm-1
           2 <= ifwhm <= 2.14748e+09
-h1        (float)   Sensor altitude in km above ground level
           unbounded
-gndalt    (float)   Ground altitude in km above sea level
           unbounded
-h2        (float)   Target altitude in km above ground level
           unbounded
-angle     (float)   Target zenith angle in degrees (180 = downlooking)
           unbounded

[-i]       (infile)  trigger input to sync execution
           (default = {none})
[-salbf]   (float)   Surface albedo
           (unbounded, default = 0.2)
[-salbp]   (cycle)   Albedo model index
         0  "Snow Cover",
         1  "Forest",
         2  "Farm",
         3  "Desert",
         4  "Ocean",
         5  "Cloud Deck"
         [default: 0 "Snow Cover"]
```

A.3. atpatrra — *Atmospheric path transmission and radiance*

Object Information

Category:	*IBSM*	Subcategory:	*Atmosphere*
Operator:	*Path Trans/Rad*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function applies the effects of atmospheric path transmission and radiance (scattered and radiated) globally to a radiance image. The function executes MODTRAN in conjunction with path data contained in the MODTRAN path data file. This file can be generated using the "MODTRAN Input" tool. It is important that the wavelength limits specified in the MODTRAN path data file cover each spectral band specified in the input image attributes. The spatial scaling attributes in the input image are

ignored.

Command Line Arguments

```
Usage for atpatrra: Atmospheric path transmission and radiance
% atpatrra
  -i          (infile)  Name input IBSM radiance image file
  -o          (outfile) Name of output IBSM radiance image file
  -ipath      (infile)  Name of input ASCII MODTRAN path data file

  [-workdir] (string)  Optional working directory for MODTRAN
             (default = (none))
```

A.4. atturbotf — *OTF due to long path atmospheric turbulence*

Object Information

Category:	*IBSM*	Subcategory:	*Atmosphere*
Operator:	*Turbulence OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies a long path turbulence OTF to a spatial spectrum. The turbulence magnitude is specified by the imaging geometry, vertical index structure parameter model, and reference index structure parameter at h=1m. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for atturbotf: OTF due to long path atmospheric turbulence
% atturbotf
  -i          (infile)  Name of input IBSM spatial spectrum file
  -o          (outfile) Name of the output IBSM spatial spectrum file
  -h1         (float)   Target altitude in km
               unbounded
  -h2         (float)   Sensor altitude in km
               unbounded
  -zenith     (float)   Target zenith angle (downlooking = 180 deg)
               unbounded
  -model      (cycle)   Vertical index structure parameter model
           0   "Simple",
           1   "Hufnagel",
           2   "Empirical"

  -v          (float)   Upper atmosphere wind speed in m/s (Hufnagel model)
               unbounded
  -cn2_1m     (float)   Reference index structure parameter at h = 1m
               cn2_1m >= 0.0
  -vrel       (float)   Sensor velocity relative to the atmosphere in m/s
```

```
            unbounded
  -td       (float)   Integration time in msec
            unbounded
  -d        (float)   Aperture diameter in cm
            d >= 0.0

  [-r0atm]  (float)   Path correlation diameter in cm (override)
            (r0atm >= 0.0, default = 0)
```

B. Data Link Man Pages

B.1. dlchannel — *Simulates data channel errors*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*Channel Errors*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function inputs an IBSM format image and simulates passing it through a noisy channel. It then outputs the resulting corrupted image.

Command Line Arguments

```
Usage for dlchannel: Simulates data channel errors
% dlchannel
  -i          (infile)  Input file that will be corrupted by the channel simulation
  -o          (outfile) Output file which contains the corrupted data by the channel simulati
  -pf         (double)  The probability that a zero will be changed to a one, or a one will b
              0 <= pf <= 1
  -pi         (double)  The probability that a bit inserted at any particular position. This
              0 <= pi <= 1
  -pd         (double)  The probability that any particular bit will be deleted.
              0 <= pd <= 1
  -pb         (double)  The probability that a burst error will start at any particular bit.
              0 <= pb <= 1
  -mb         (double)  The mean length of the error bursts. Together with the Distribution c
              mb >= 0.0
  -db         (cycle)   Determines whether the burst errors have constant or variable length
            0  "Constant Length",
            1  "Variable Length"

  -tb         (cycle)   If this is "zero" or "one" then all of the bits in a burst error will
            0  "Zeros",
            1  "Ones",
            2  "Zeros/Ones"
```

```
-hdrlen    (integer) Defines the number of bits at the beginning of each band that are gua
           hdrlen >= 0
```

B.2. dlbuffer — *Simulates the effects of buffering on input data*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*Finite Buffer*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function simulates the buffering of an IBSM format image. It inputs an image and its corresponding bits/block image (see the dljpegcomp man page for more information). The latter is produced automatically by IBSM compressor glyphs, or it may be produced manually for non-compressed images. Its outputs consist of the buffered data, which may be different from the the original data if overflows and underflows occurred. It also produces a graphable data file that represents how many bits were in the buffer at each time step.

Command Line Arguments

```
Usage for dlbuffer: Simulates the effects of buffering on input data
% dlbuffer
  -i         (infile)  Input IBSM compressed file to buffer
  -ibits     (infile)  Input IBSM bits/block file from compression output
  -p_in      (double)  The time between element arrivals.
             p_in > 0.0
  -p_out     (double)  The time between element departures.
             p_out > 0.0
  -mode      (cycle)   Configures the buffer to operate as if it was either on the input or
          0  "Input Bit/Output Block",
          1  "Input Block/Output Bit"

  -buf_s     (integer) The total amount of bits the buffer can hold.
             buf_s >= 0
  -sim_un    (cycle)   Turns on or off the simulation of buffering underflow.
          0  "Off",
          1  "On"

  -sim_ov    (cycle)   Turns on or off the simulation of buffering overflow.
          0  "Off",
          1  "On"

  -delay     (double)  The number of seconds that the buffer is allowed to fill before the f
             delay >= 0.0
  -ilev      (integer) In effect the buffer is filled with this many dummy bits before it st
             ilev >= 0
  -hdr_len   (integer) Number of bits protected from buffer corruption
             hdr_len >= 0
```

```
[-o]        (outfile) Resulting output data file after buffering
            (default = {none})
[-obof]     (outfile) Holds a one-dimensional vector filled with the number of bits in the
            (default = {none})
[-step_sz]  (double)  If secified, the buffer occupancy file is updated once every step siz
            (unbounded, default = -1)
```

B.3. dljpegcomp — *Performs NITFS JPEG Compression*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*JPEG Compress*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function inputs an IBSM format image, an image quality factor, and a restart interval length. It compresses each band separately and returns both the compressed data and, optionally, a bits/block image.

Glossary

Bits/Block Image (BB image): This is an IBSM term. The BB image for some arbitrary image T has one element for each MCU in T. This element contains the number of bits in the compressed representation of that MCU.

The BB image is derived by counting the number of bits in each restart interval and averaging this over the number of MCU's in a restart interval. The BB image is set equal to this average for the entire restart interval. Therefore the shorter the restart interval length, the more accurate the BB image. And (nearly) perfectly accurate BB images are produced if and only if the restart interval is one MCU long.

Image Quality Factor (IQF): An NJC and IBSM term which denotes a number from 1 to 5 that controls the accuracy with which the image is reconstructed. Lower values mean that fewer bits are used in the compressed file. Unfortunately they also mean that the reconstructed image is cruder. This range of values (i.e. 1 to 5) was set by p. 60 of MIL-STD-188-198A (and also page 40 of MIL-STD-2500A), but is not part of the JPEG standard itself.

JPEG: JPEG stands for Joint Photographic Experts Group, the name of the committee that came up with a standardized method for image compression. More information about it can be found at: http://www.cis.ohio-state.edu/hypertext/faq/usenet/jpeg-faq/top.html from the book: W. Peenebaker, J. Mitchell, JPEG_Still_Image_Data_Compression_ Standard, Van Nostrand Reinhold, N.Y., 1993 or from MIL-STD-188-198A.

MCU: MCU is a JPEG term which stands for Minimum Coded Unit. To define it, start with any image and divide it into 8x8 blocks which cover the whole image but do not overlap. Then each of these blocks is an MCU.

NJC: NJC stands for the NITFS JPEG Compression Software Library. These are some routines that were written by the Optivision Co. (1477 Drew Ave.. Suite 102. Davis, CA 95616) to provide an

example of software that implements MIL-STD-188-198. Release 2.2i is used here. This library was chosen over its competitors partly because it may eventually provide unique features (such as the use of forward error correcting codes for data protection) that may be of interest to users of IBSM.

Restart Interval: This is a JPEG term which denotes a collection of one or more consecutive MCU's. After the compressed data for each restart interval, the compressor transmits a code. This may allow the receiver to regain synchronization when a bit error occurred in the compressed data for that restart interval. In other words, channel errors should be confined to the restart interval in which they occurred. So for optimal error protection we want the restart intervals as short as possible. The tradeoff is that each restart code takes about 20 bits to transmit, so the shorter the restart interval the more bits are spent on overhead. To comply with paragraph 5.2.3.3.1.4 of MIL-STD-188-198A, an image's restart interval must be no larger than the number of MCU's it takes to span the width of that image.

Command Line Arguments

```
Usage for dljpegcomp: Performs NITFS JPEG Compression
% dljpegcomp
  -i          (infile)  The name of the file that will be compressed
  -ocf        (outfile) Special format IBSM file that contains compressed JPEG data
  -iqf        (integer) Image quality factor where 5 is the highest quality and 1 is the lowe
               1 <= iqf <= 5
  -rilen      (integer) MCU=Minimum Coded Unit (One 8x8 block): The number of 8x8 image block
               rilen > 0

  [-obf]      (outfile) An integer image each of whose elements contain the number of bits pe
              (default = {none})
```

B.4. dljpegdecomp — *JPEG decompression of the input image*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*JPEG Decompress*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function inputs an IBSM format image created with the dljpegcomp function with possibly post-processing by, for example, the dlbuffer or dlchannel functions. It then creates the corresponding decompressed image. See the dljpegcomp man page for more information.

Command Line Arguments

```
Usage for dljpegdecomp: JPEG decompression of the input image
% dljpegdecomp
  -i          (infile)  IBSM formatted file that was compressed with the JPEG Compression gly
  -o          (outfile) Decompressed JPEG image file
```

B.5. dltsvq — *TSVQ - Tree-Structured Vector Quantization Compression and Decompression*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*TSVQ Comp/Decomp*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function has two modes. The first reads an input image and a TSVQ codebook. It compresses the input using the codebook and produces both an image of indices and a bits/(unit time) image (suitable for the buffer glyph). Also, the restart interval setting determines the distance between consecutive restart markers in the image of indices, unless it's zero in which case restart markers are not used.

The second mode reads an image of indices and outputs thedecompressed image.

Command Line Arguments

```
Usage for dltsvq: TSVQ - Tree-Structured Vector Quantization Compression and Decompression
% dltsvq
  -i         (infile)  File name of the original image or compressed image depending on the
  -o         (outfile) File name of the output decompressed image or output compressed image
  -dtype     (integer toggle) Input data type that defines whether the input/original file :
           1  (8 Bit),
           2  (12 Bit),
           or 3  (Float)

  -mode      (integer toggle) Selects the mode of operation as either compress or decompress
           1  (Compress),
           or 2  (Decompress)

  -ri_len    (integer) After every ri_len consecutive indices the compressor will insert a r
              ri_len >= 0

  [-bits]    (outfile) Optional output file that represents a mesh whose interstices are 8x8
             (default = {none})
  [-stem]    (string)  Codebook file names stem or root where the actual file names are ster
             (default = {none})
```

B.6. dlwavelet — *Wavlet Compression and Decompression*

Object Information

Category:	*IBSM*	Subcategory:	*Data Link*
Operator:	*Wavelet Comp/Decomp*	Object Type:	*script*
In Cantata:	*Yes*		

Description

Wavelet coding tries to approximate the compression performance of very high-dimensional vector quantization at a reasonable complexity. The wavelet decoder takes the indices that the VQ produced and reconstructs the image in the transformed coordinate system. Finally, it returns the image to the normal coordinate system by multiplying by the inverse wavelet transform.

Command Line Arguments

```
Usage for dlwavelet: Wavlet Compression and Decompression
% dlwavelet
  -i          (infile)  If mode is "c" this should contain a float-format input image whose d
  -o          (outfile) If nc is 1, this is the name of a file that will contain wavelet tran

  [-stem]     (string)  The prefix of two codebook file names: one ends in "_shape.viff", whi
              (default = {none})
  [-mode]     (string)  Wavelet mode.
              (default = c)
  [-ril]      (integer) The restart interval length.
              (ril >= 0, default = 0)
  [-lev]      (integer) The number of levels in the wavelet decomposition.
              (lev > 0, default = 1)
  [-nc]       (integer toggle) Output training data
            1  (No),
            or 2  (Yes)
[default: 1]
```

C. Detector Man Pages

C.1. decteotf — *OTF due to charge transfer inefficiency*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*CTE OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF for the effects of charge transfer inefficiency (CTE) in charge-coupled device (CCD) detector readout. The per transfer CTE is assumed identical in the x and y directions. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for decteotf: OTF due to charge transfer inefficiency
% decteotf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -px      (float)   Detector X pitch in microns
           px >= 0.0
  -py      (float)   Detector Y pitch in microns
           py >= 0.0
  -f       (float)   Sensor focal length in cm
           f >= 0.0
  -ntx     (integer) Number of transfers in the  X direction
           ntx > 0
  -nty     (integer) Number of transfers in the Y direction
           nty > 0
  -phase   (integer) Number of clock phases
           phase > 0
  -eps     (float)   Charge transfer efficiency (per phase)
           eps >= 0.0
```

C.2. dedetect — *Photoelectric conversion at the detector plane*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Detection*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function converts an input focal plane irradiance image into a noiseless detected photoelectron image based on detector parameters.

Command Line Arguments

```
Usage for dedetect: Photoelectric conversion at the detector plane
% dedetect
  -i       (infile)  Name of input IBSM irradiance image file
  -o       (outfile) Name of output IBSM photoelectron image file
  -eta     (float)   Detector mean quantum efficiency over band
           unbounded
  -ntdi    (integer) Number of TDI stages in use
           ntdi > 0
  -agx     (integer) Aggregation in X direction
           agx > 0
  -agy     (integer) Aggregation in Y direction
           agy > 0
  -wx      (float)   Detector X width in microns
           wx >= 0.0
  -wy      (float)   Detector Y width in microns
           wy >= 0.0
```

```
-td     (float)   Detector dwell (integration) time in msec
          td >= 0.0
-jd     (float)   Dark current density in microamps/cm2
          jd >= 0.0
-sat    (float)   Saturation photoelectron level (full well) in electrons
          sat > 0.0
```

C.3. dediffotf — *OTF due to CCD charge diffusion*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Diffusion OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF for the effects of carrier diffusion in a charge coupled device (CCD) as a function of material properties and dimensions. Note that the absorption coefficient, at least in the case of silicon, is wavelength dependent. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for dediffotf: OTF due to CCD charge diffusion
% dediffotf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of the output IBSM spatial spectrum file
  -alpha   (float)   CCD material absorption coefficient in mm-1
             alpha >= 0.0
  -ald     (float)   CCD depletion layer width in mm
             ald >= 0.0
  -alo     (float)   Carrier diffusion length in mm
             alo >= 0.0
  -f       (float)   Sensor focal length in cm
             f >= 0.0
```

C.4. denoise — *Addition of detector and electronic noise*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Noise*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function adds a random noise process to a photoelectron image. Two types of noise are modeled. The first is signal dependent, Poisson distributed shot noise, which can be either enabled or disabled. The second is signal independent, additive noise that is assumed zero mean, gaussian distributed with a specified root mean square (RMS) level. The noise sources are assumed both jointly and pixel-to- pixel independent.

Command Line Arguments

```
Usage for denoise: Addition of detector and electronic noise
% denoise
  -i       (infile)  Name of input IBSM photoelectron image file
  -o       (outfile) Name of output IBSM photoelectron image file
  -shot    (cycle)   Shot noise selection switch
           0  "No",
           1  "Yes"

  -sigma   (float)   RMS additive noise in electrons
            sigma >= 0.0
```

C.5. denonunif — *Adds effects of random detector nonuniformity to a detected image*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Nonuniformity*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function applies the effects of detector nonuniformity to an input image. The level of nonuniformity is specified separately for gain (multiplicative) and offset (additive) portions. Both are modeled as gaussian random processes with the latter scaled relative to a user specified maximum signal level. The user can also specify an input autocorrelation file which characterizes the spatial dependencies. If not specified, the nonuniformity is assumed pixel-to-pixel independent.

Command Line Arguments

```
Usage for denonunif: Adds effects of random detector nonuniformity to a detected image
% denonunif
  -i        (infile)  Input IBSM image file name
  -o        (outfile) Output IBSM image file name
  -gain     (float)   Percent gain uniformity
             gain >= 0.0
  -offset   (float)   Percent offset nonuniformity
             offset >= 0.0
  -nmax     (float)   Maximum signal level
             nmax >= 0.0
```

```
[-rho]    (infile)  Input IBSM autocorrelation file name
          (default = (none))
```

C.6. dequantiz — *Scalar image quantization*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Quantization*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function quantizes a detected photoelectron image through a scalar roundoff process to 2**b digitization levels linearly distributed between the user-specified minimum and maximum levels.

Command Line Arguments

```
Usage for dequantiz: Scalar image quantization
% dequantiz
  -i      (infile)  Name of input IBSM photoelectron image file
  -o      (outfile) Name of output IBSM photoelectron image file
  -nmin   (float)   Minimum quantization level in electrons
          nmin >= 0.0
  -nmax   (float)   Maximum quantization level in electrons
          nmax >= 0.0
  -bits   (integer) Digitization levels as power of 2
          bits > 0
```

C.7. desamplin — *Focal plane image sampling*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Sampling*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function performs a focal plane sampling of an input image by an array of point detectors, assuming detector width effects have been previously incorporated. The resampling process is achieved using a bi-linear interpolation of the input image data with user specified absolute alignment of the sampling grid to the input image. This alignment is specified by way of the fractional (relative to sample spacing) offsets.

Command Line Arguments

```
Usage for desamplin: Focal plane image sampling
% desamplin
  -i      (infile)  Name of input IBSM image file
  -o      (outfile) Name of output IBSM image file
  -px     (float)   Detector X pitch in microns
           px >= 0.0
  -py     (float)   Detector Y pitch in microns
           py >= 0.0
  -agx    (integer) Aggregation in X direction
           agx > 0
  -agy    (integer) Aggregation in Y direction
           agy > 0
  -f      (float)   Sensor focal length in cm
           f >= 0.0
  -xoff   (float)   Fractional sampling offset in X direction
           unbounded
  -yoff   (float)   Fractional sampling offset in Y direction
           unbounded
```

C.8. desizeotf — *OTF due to finite detector width*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*Detector Size OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF from the finite detector size to an input spatial spectrum. The detector is assumed to have a rectangular form. Spatial scaling (rad-1 units) and spectral and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for desizeotf: OTF due to finite detector width
% desizeotf
  -i      (infile)  Name of input IBSM spatial spectrum file
  -o      (outfile) Name of output IBSM spectrum file
  -dx     (float)   Detector X width in microns
           dx >= 0.0
  -dy     (float)   Detector Y width in microns
           dy >= 0.0
  -px     (float)   X detector pitch in microns
           px > 0.0
  -py     (float)   Y detector pitch in microns
           py > 0.0
  -agx    (integer) Aggregation in X direction
```

```
        agx > 0
-agy    (integer) Aggregation in Y direction
        agy > 0
-f      (float)   Sensor focal length in cm
        f >= 0.0
```

C.9. detdiotf — *OTF due to detector time-delay-integration*

Object Information

Category:	*IBSM*	Subcategory:	*Detector*
Operator:	*TDI OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF due to the detector time-delay-integration (TDI) process to an input spatial spectrum. The TDI can occur in either the x or y directions, and may exhibit a mismatch between the TDI clocking rate and the line-of-sight (LOS) motion rate. Even for a perfect rate match (beta=1.0), an OTF effect occurs due to the LOS drift of one detector phase. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for detdiotf: OTF due to detector time-delay-integration
% detdiotf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -dir     (cycle)   Direction of TDI (X or Y)
           0  "X",
           1  "Y"

  -d       (float)   Detector pitch in TDI direction in microns
           d >= 0.0
  -f       (float)   the value of focal length in cm
           f >= 0.0
  -ntdi    (integer) Number of TDI stages in use
           ntdi > 0
  -beta    (float)   TDI rate match = TDI rate / LOS motion rate
           beta >= 0.0
  -phase   (integer) Number of clock phases
           phase > 0
```

D. Optics Man Pages

D.1. opdefootf — *OTF due to Gaussian optical blur function*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Defocus OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies the OTF due to optics defocus to an input spatial spectrum. The defocus (or blur) function is modeled as a gaussian distribution for which the full-width, half maximum (FWHM) in angular space is specified independently for the X and Y axes. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for opdefootf: OTF due to Gaussian optical blur function
% opdefootf
  -i    (infile)  Name of input IBSM spatial spectrum file
  -o    (outfile) Name of output IBSM spatial spectrum file
  -wx   (float)   FWHM X width of blur function in mrad
         wx >= 0.0
  -wy   (float)   FWHM Y width of blur function in mrad
         wy >= 0.0
```

D.2. opdiffotf — *OTF due to aperture diffraction*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Diffraction OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies a diffraction limited OTF to an input spatial spectrum for a circular or rectangular aperture. The relative obscuration applies only to a centrally obscured circular aperture, and is the ratio of obscuration to aperture diameters. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be

produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for opdiffotf: OTF due to aperture diffraction
% opdiffotf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -shape   (cycle)   Aperture shape description
          0  "Circular",
          1  "Rectangular"

  -dx      (float)   Aperture diameter (circular) or X width (rectangular) in cm
           dx >= 0.0
  -dy      (float)   Aperture Y width (rectangular only) in cm
           dy >= 0.0
  -obs     (float)   Ratio of obscuration to aperture diameter (circular only)
           unbounded
```

D.3. opdrifotf — *OTF due to linear line-of-sight drift*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Drift OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF due to a linear angular drift of the sensor line-of-sight during the integration time to an input spatial spectrum. The magnitude of the linear drift is specified independently for the x and y directions. Only the effect of drift on the OTF is modeled; possible image distortion effects are not. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for opdrifotf: OTF due to linear line-of-sight drift
% opdrifotf
  -i    (infile)  Name of input IBSM spatial spectrum file
  -o    (outfile) Name of output IBSM spatial spectrum file
  -ax   (float)   Angular X drift in mrad
        ax >= 0.0
  -ay   (float)   Angular Y drift in mrad
        ay >= 0.0
```

D.4. opjittotf — *OTF due to high frequency line-of-sight jitter*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Jitter OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF due to high frequency (with respect to the sensor integration time) jitter of an optical sensor line-of-sight. By virtue of this high bandwidth assumption, the jitter is specified by its x and y axis root-mean square (RMS) components and is modeled with a zero-mean, gaussian assumption. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for opjittotf: OTF due to high frequency line-of-sight jitter
% opjittotf
  -i      (infile)  Name of input IBSM spatial spectrum file
  -o      (outfile) Name of output IBSM spatial spectrum file
  -sigx   (float)   RMS X jitter in mrad
           sigx >= 0.0
  -sigy   (float)   RMS Y jitter in mrad
           sigy >= 0.0
```

D.5. opradiom — *Conversion of pupil plane radiance to focal plane irradiance*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Radiometry*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function performs a radiometric conversion of an effective pupil plane radiance image to a focal plane irradiance image. The model accommodates sensor throughput, thermal radiation of the optical chain, and stray radiance. The input data must be properly scaled in radiance (W/m2sr) units.

Command Line Arguments

```
Usage for opradiom: Conversion of pupil plane radiance to focal plane irradiance
% opradiom
```

```
-i       (infile)  Name of input IBSM radiance image file
-o       (outfile) Name of output IBSM irradiance image file
-fnum    (float)   Effective sensor F-number
          fnum > 0.0
-trans   (float)   Mean optical transmission of sensor over band
          unbounded
-em      (float)   Effective emissivity of sensor optics over band
          unbounded
-temp    (float)   Effective temperature of sensor optics in K
          temp >= 0.0
-stray   (float)   Effective sensor stray radiance in uW/cm2sr
          stray >= 0.0
```

D.6. opuserotf — *OTF defined by 1-D user data*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*User OTF (1-D)*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF based on user data to an input spatial spectrum. The data is contained in an ASCII file in formats accommodating real or complex data specified with radial form (radial), with separability in the x and y axes (x or y), or with separability and similarity in the x and y axes (x and y). The user data is interpolated using linear or spline techniques to the input spatial spectrum grid prior to forming the 2-D OTF. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool. The input file contains a series of lines, one for each OTF point, in the following format:

Radial,real: Frequency OTF

Radial,complex: Frequency OTFreal OTFimag

X or Y,real: Frequency OTF

X or Y,complex: Frequency OTFreal OTFimag

X and Y,real: FrequencyX OTFX FrequencyY OTFY

X and Y,complex: FrequencyX OTFXreal OTFXimag FrequencyY OTFYreal OTFYimag

Command Line Arguments

```
Usage for opuserotf: OTF defined by 1-D user data
% opuserotf
  -i       (infile)  Name of input IBSM spatial spectrum file
```

```
-o       (outfile) Name of output IBSM spatial spectrum file
-fname   (infile)  Name of ASCII file containing user OTF data
-type1   (cycle)   User OTF data format
         0  "Radial",
         1  "X or Y",
         2  "X and Y"

-type2   (cycle)   User OTF data type
         0  "Real",
         1  "Complex"

-type3   (cycle)   Interpolation type
         0  "Linear",
         1  "Spline"
```

D.7. opusrotf2 — *OTF defined by 2-D user data in OTF or PSF form*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*User OTF (2-D)*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF based on bilinear interpolation of a user 2-D OTF or discrete Fourier transformation of a user 2-D PSF. The output data format (e.g. real vs. complex) is defined by the input spatial spectrum file, which must contain spatial scaling (rad-1 units) attributes. Normalization produces a unity OTF at zero frequency.

Command Line Arguments

```
Usage for opusrotf2: OTF defined by 2-D user data in OTF or PSF form
% opusrotf2
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -otf     (infile)  Name of ASCII file containing user OTF or PSF data
  -type    (cycle)   File type
           0  "PSF",
           1  "OTF"

  -norm    (cycle)   Normalization switch
           0  "No",
           1  "Yes"

  -bands   (cycle)   OTF/PSF bands
           0  "Single",
           1  "Multiple"
```

D.8. opwaveotf — *OTF due to random pupil plane wavefront error*

Object Information

Category:	*IBSM*	Subcategory:	*Optics*
Operator:	*Wavefront OTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates and applies an OTF due to a random wavefront error specified in the sensor pupil plane to an input spatial spectrum. The statistics of the wavefront error are modeled as zero mean, gaussian with independently specified correlation lengths for the x and y axes in the pupil plane. Spatial scaling (rad-1 units) and band attributes must be available for the input image, and the data must correspond to a spatial spectrum, which can be produced from an image using the "Transform" tool.

Command Line Arguments

```
Usage for opwaveotf: OTF due to random pupil plane wavefront error
% opwaveotf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -sigma   (float)   RMS wavefront error in waves
            sigma >= 0.0
  -wref    (float)   Reference wavelength in microns
            wref >= 0.0
  -lx      (float)   Wavefront error X correlation length in cm
            lx >= 0.0
  -ly      (float)   Wavefront error Y correlation length in cm
            ly >= 0.0
```

E. Performance Man Pages

E.1. pecremtf — *Generate 1-D MTF from 2-D MTF*

Object Information

Category:	*IBSM*	Subcategory:	*Performance*
Operator:	*Create MTF*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates a 1D MTF based on a real-valued 2D spatial spectrum (MTF) file. The

generated MTF can lie along either the X or Y axes of the input data, and is placed as a single row in an IBSM output file. The frequency scale (min/max) can be either the same as the input (automatic) or overriden (fixed). In the latter case , the data is linearly interpolated to the specified grid. This generated IBSM file is of a form suitable for input to the "Sensor Performance" function.

Command Line Arguments

```
Usage for pecremtf: Generate 1-D MTF from 2-D MTF
% pecremtf
  -i       (infile)  Name of input IBSM spatial spectrum file
  -o       (outfile) Name of output IBSM spatial spectrum file
  -axis    (cycle)   Designator of MTF axis
           0  "X",
           1  "Y"

  -scale   (cycle)   Designator for spatial frequency scaling method
           0  "Auto",
           1  "Fixed"


  [-fmin]  (float)   Minimum spatial frequency (fixed scale)
           (fmin >= 0.0, default = 0)
  [-fmax]  (float)   Maximum spatial frequency (fixed scale)
           (fmax >= 0.0, default = 100000)
  [-nfreq] (integer) Number of spatial frequency points (fixed scale)
           (nfreq > 0, default = 100)
```

E.2. pecrespec — *Generation of atmospheric and sensor spectral distributions*

Object Information

Category:	*IBSM*	Subcategory:	*Performance*
Operator:	*Create Spectra*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates an eight-band IBSM file containing atmospheric and sensor spectral distributions. The bands contain, respectively, (0) atmospheric path transmission, (1) atmospheric path radiance, (2) irradiance at the target fro a direct source (sun or moon), (3) diffuse downwelling radiance at the target, (4) target reflectance, (5) background reflectance, (6) sensor optical chain transmission, and (7) detector quantum efficiency. All are in single row format of "nwavl" columns. When the override parameters are not selected, the first four distributions (0-3) are computed via MODTRAN in reference to the ASCII path data file, the following two (4-5) from the ASCII reflectance file data, and the final two (6-7) from the ASCII sensor file data. In all cases, the spectral distributions are computed through a linear interpolation of the data, and can be overridden with the spectrally flat override parameters. The minimum and maximum x parameters in the IBSM header contain the wavelength minimum and maximum in meters. The minimum y, maximum y, center wavelengths, and spectral bandwidths contained in the header should be ignored. This generated IBSM file is of a form suitable for input to the "Sensor Performance" function.

Command Line Arguments

```
Usage for pecrespec: Generation of atmospheric and sensor spectral distributions
% pecrespec
  -o              (outfile) Name of output IBSM spectral distribution file
  -wmin           (float)   Minimum wavelength of distributions in microns
                   unbounded
  -wmax           (float)   Maximum wavelength of distributions in microns
                   unbounded
  -nwavl          (integer) Number of spectral points in the distributions
                   nwavl > 0

  [-i]            (infile)  trigger input to sync execution
                  (default = {none})
  [-workdir]      (string)  Optional working directory for MODTRAN
                  (default = {none})
  [-fname_atm]    (infile)  Name of the input MODTRAN path data file
                  (default = {none})
  [-trans]        (float)   Atmospheric path transmission (override)
                  (unbounded, default = 1)
  [-pathrad]      (float)   Atmospheric path radiance in uW/cm2umsr (override)
                  (pathrad >= 0.0, default = 0)
  [-direct]       (float)   Direct Source irradiance at the target in uW/cm2um (override)
                  (direct >= 0.0, default = 0)
  [-zenith]       (float)   Source zenith angle in degrees (used with direct irradiance)
                  (unbounded, default = 0)
  [-diffuse]      (float)   Diffuse downwelling radiance in uW/cm2umsr (override)
                  (diffuse >= 0.0, default = 0)
  [-fname_refl]   (infile)  Name of ASCII reflectance data file
                  (default = {none})
  [-nrefl]        (integer) Number of lines in the reflectance data file
                  (2 <= nrefl <= 2.14748e+09, default = 2)
  [-rtgt]         (float)   Target reflectance (override)
                  (unbounded, default = 1)
  [-rbkg]         (float)   Background reflectance (override)
                  (unbounded, default = 0)
  [-fname_sens]   (infile)  Name of the ASCII sensor data file
                  (default = {none})
  [-nsens]        (integer) Number of lines in the sensor data file
                  (2 <= nsens <= 2.14748e+09, default = 2)
  [-topt]         (float)   Optical chain transmission (override)
                  (unbounded, default = 1)
  [-eta]          (float)   Detector quantum efficiency (override)
                  (unbounded, default = 1)
```

E.3. peexpctl — *Exposure Control*

Object Information

Category:	*IBSM*	Subcategory:	*Performance*
Operator:	*Exposure Control*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function performs exposure control by determining the pixel dwell time, TDI stages, and aggregation levels which place the maximum of the target or background signal levels between specified mnimum and maximum levels. For the signal computation, sensor spectral distributions using the "Create Spectra" function are used. The output consists of sensor configuration values which are placed in a 5 band floating-point VIFF file (one value per band), which canbe extracted to global variables using the "Print Value" function. The output file format is as follows:

Band 0	Pixel dwell time in msec
Band 1	TDI stages
Band 2	X aggregation
Band 3	Y aggregation
Band 4	Signal level (max) in electrons

Command Line Arguments

```
Usage for peexpctl: Exposure Control
% peexpctl
  -i_spect    (infile)  Name of input IBSM spectral distribution file
  -o          (outfile) Name of output VIFF result file
  -dwell      (float)   Desired pixel dwell time in msec
               dwell >= 0.0
  -tdi        (string)  Allowable TDI stages (input as string with blank separators)
  -agx        (string)  Allowable X aggregations (input as string with blank separators)
  -agy        (string)  Allowable Y aggregation (input as string with blank separators)
  -range      (float)   Target slant range in km
               range > 0.0
  -ttgt       (float)   Target temperature in Kelvin
               ttgt >= 0.0
  -tbkg       (float)   Background temperature in Kelvin
               tbkg >= 0.0
  -f          (float)   Sensor focal length in cm
               f >= 0.0
  -fnum       (float)   Effective sensor F-number
               fnum >= 0.0
  -eopt       (float)   Effective emissivity of sensor optics over band
               unbounded
  -topt       (float)   Effective temperature of sensor optics in Kelvin
               topt >= 0.0
  -lstray     (float)   Effective sensor stray radiance in uW/cm2sr
               lstray >= 0.0
  -wx         (float)   Detector X width in microns
               wx >= 0.0
  -wy         (float)   Detector Y width in microns
               wy >= 0.0
  -jd         (float)   Detector dark current density in microamps/cm2
               jd >= 0.0
  -nmin       (float)   Minimum signal level for exposure control in electrons
               nmin >= 0.0
  -nmax       (float)   Maximum signal level for exposure control in electrons
               nmax > 0.0
```

E.4. peimgqual — *Computes edge response functions and NIIRS from test image*

Object Information

Category:	*IBSM*	Subcategory:	*Performance*
Operator:	*Image Quality*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function computes image quality metrics including edge response functions, GSD, SNR, and NIIRS from an image containing either test patterns or edge features. The user specifies the locations of horizontal and vertical edges as well as uniform image regions for noise estimation. Outputs include ASCII files containing the X and Y edge response functions (Two column format) and/or a metric summary file.

Command Line Arguments

```
Usage for peimgqual: Computes edge response functions and NIIRS from test image
% peimgqual
  -i        (infile)  Input IBSM image file name
  -xve      (float)   X position of vertical edge in mrad
            unbounded
  -yve      (float)   Y position of vertical edge in mrad
            unbounded
  -hve      (float)   Vertical edge size in mrad
            unbounded
  -xhe      (float)   X position of horizontal edge in mrad
            unbounded
  -yhe      (float)   Y position of horizontal edge in mrad
            unbounded
  -hhe      (float)   Horizontal edge size in mrad
            unbounded
  -xtr      (float)   X position of target region in mrad
            unbounded
  -ytr      (float)   Y position of target region in mrad
            unbounded
  -xbr      (float)   X position of background region in mrad
            unbounded
  -ybr      (float)   Y position of background region in mrad
            unbounded
  -vrs      (float)   Vertical region size in mrad
            unbounded
  -hrs      (float)   Horizontal region size in mrad
            unbounded
  -range    (float)   Nominal slant range in km
            range > 0.0

  [-o1]     (outfile) Output ASCII X response file name
            (default = {none})
  [-o2]     (outfile) Output ASCII Y response file name
            (default = {none})
  [-o3]     (outfile) Output ASCII results file name
```

```
(default = (none))
```

E.5. pesensper — *Compute sensor performance metrics*

Object Information

Category:	*IBSM*	Subcategory:	*Performance*
Operator:	*Sensor Performance*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function computes metrics characterizing sensor performance (including SNR, GRD, and NIIRS) based on a set of sensor parameters; target, background, atmosphere, and sensor spectral distributions using the "Create Spectra" function; and x and y axis MTF functions using the "Create MTF" function. The metric is written to an ASCII file along with an arbitrary independent variable. Depending on the 'update' parameter, the file is initialized before writing or the data is appended to an existing file. In the long form output data format, the full input parameter set is written to the file upon initialization (update = 0), and the metric data is augmented with additional intermediate computation results. The short form output data format results in a two column ASCII file containing the independent variable and computed metric for each update execution, which is useful for generating parametric plot data. The SNR metric refers to the target to background difference signal relative to the detected noise level. In this case, the axis is ignored. The GRD metric is provided in meters and is given in a plane orthogonal to the sensor line-of-sight axis. The image quality metric (NIIRS) is also computed relative to an orthogonal plane ground sample distance (GSD).

Command Line Arguments

```
Usage for pesensper: Compute sensor performance metrics
% pesensper
  -i_spect    (infile)  Name of input IBSM spectral distribution file
  -o          (outfile) Name of output ASCII result file
  -metric     (cycle)   Sensor performance metric
          0   "Signal to Noise Ratio",
          1   "Ground Resolved Distance",
          2   "Image Quality"

  -wx         (float)   Detector X width in microns
              wx >= 0.0
  -xvar       (float)   Independent variable for plots
              unbounded
  -wy         (float)   Detector Y width in microns
              wy >= 0.0
  -update     (integer) Output file update switch (0 = new file, 1 = update)
              0 <= update <= 1
  -px         (float)   Detector X pitch in microns
              px >= 0.0
  -format     (cycle)   Output data format
          0   "Short Form",
          1   "Long Form"
```

```
-py         (float)   Detector Y pitch in microns
            py >= 0.0
-td         (float)   Detector dwell (integration) time in msec
            td >= 0.0
-range      (float)   Target slant range in km
            range > 0.0
-jd         (float)   Detector dark current density in microamps/cm2
            jd >= 0.0
-ttgt       (float)   Target temperature in Kelvin
            ttgt >= 0.0
-ntdi       (integer) Number of detector TDI stages
            ntdi > 0
-tbkg       (float)   Background temperature in Kelvin
            tbkg >= 0.0
-agx        (integer) Aggregation in X direction
            agx > 0
-agy        (integer) Aggregation in Y direction
            agy > 0
-nmin       (float)   Quantizer minimum level in electrons
            nmin >= 0.0
-f          (float)   Sensor focal length in cm
            f >= 0.0
-nmax       (float)   Quantizer maximum level in electrons
            nmax > 0.0
-fnum       (float)   Effective sensor F-number
            fnum >= 0.0
-eopt       (float)   Effective emissivity of sensor optics over band
            unbounded
-bits       (integer) Digitization levels as power of 2
            bits > 0
-topt       (float)   Effective temperature of sensor optics in Kelvin
            topt >= 0.0
-sigma      (float)   Additive noise level RMS in electrons
            sigma >= 0.0
-lstray     (float)   Effective sensor stray radiance in uW/cm2sr
            lstray >= 0.0
-gain       (float)   Processing noise gain
            gain > 0.0

[-i_mtfx]   (infile)  Name of input IBSM MTF file (X axis)
            (default = {none})
[-i_mtfy]   (infile)  Name of input IBSM MTF file (Y axis)
            (default = {none})
[-agc]      (float)   Automatic gain correction level in electrons (optional)
            (agc > 0.0, default = 1e+07)
```

F. Processing Man Pages

F.1. printerlace — *Interlaces IBSM images*

Object Information

Category:	*IBSM*	Subcategory:	*Processing*
Operator:	*Image Interlace*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function interlaces up to four images in the user specified direction: X (width), Y (height), or band (element). The interlaced images must all be the same size.

Command Line Arguments

```
Usage for printerlace: Interlaces IBSM images
% printerlace
  -i1          (infile)  First IBSM Image to Interlace (Required)
  -i2          (infile)  Second IBSM Image to Interlace (Required)
  -o           (outfile) Output IBSM Interlaced File
  -mode        (cycle)   Interlace Mode Selection
           0   "  X ",
           1   "  Y",
           2   "Bands"


  [-i3]        (infile)  Third IBSM Image to Interlace (Optional)
               (default = {none})
  [-i4]        (infile)  Fourth IBSM Image to Interlace (Optional)
               (default = {none})
```

F.2. praggreg — *Sums image elements in width, height, and bands*

Object Information

Category:	*IBSM*	Subcategory:	*Processing*
Operator:	*Aggregate*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function sums image elements in width, height, and bands to a user specified aggregation level. By virtue of the aggregation, the output image is smaller in each dimension by the aggregation factor. The residual image, when the size is not an integer multiple of the aggregation factor, is truncated. If the projection option is selected for a particular dimension, the aggregation factor is overriden by the by the dimension size. Note that overflows may occur in the summation process.

Command Line Arguments

```
Usage for praggreg: Sums image elements in width, height, and bands
% praggreg
  -i          (infile)  IBSM file used as input to the aggregation function
  -o          (outfile) Output IBSM residual image  from the aggregation function
  -xagg       (integer) User specified aggregation factor in X
               1 <= xagg <= 2.14748e+09
  -yagg       (integer) User specified aggregation factor in Y
               1 <= yagg <= 2.14748e+09
  -bagg       (integer) User specified aggregation factor in bands
               1 <= bagg <= 2.14748e+09
  -projx      (boolean) Project input image in X (Overrides X Aggregation Factor when true)
  -projy      (boolean) Project input image in Y (Overrides Y Aggregation Factor when true)
  -projb      (boolean) Project input image in bands (Overrides Band Aggregation Factor when
```

G. Utilities Man Pages

G.1. ibsminfo — *Extract IBSM header information*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*List IBSM File Data*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function generates an ASCII file containing information extracted from the IBSM header and attributes. This includes data type, image size, number of bands, spatial scaling, and the center wavelength and spectral bandwidth for each band.

Command Line Arguments

```
Usage for ibsminfo: Extract IBSM header information
% ibsminfo
  -i   (infile)  Name of input IBSM file

  [-o] (outfile) Name of output ASCII file (optional)
       (default = {none})
```

G.2. utcreplot — *Formats IBSM Data for Plotting*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*Create Plot Data*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function extracts a row or column of an IBSM file and outputs it along with spatial scaling data into two column ASCII format for plotting.

Command Line Arguments

```
Usage for utcreplot: Formats IBSM Data for Plotting
% utcreplot
  -i        (infile)  input IBSM image or spectrum
  -o        (outfile) resulting output ASCII plot file
  -band     (integer) the value of integer
             0 <= band <= 1023
  -direct   (cycle)   image direction cycle value
          0  "Row",
          1  "Column"

  -rcnum    (integer) the value of row or column number
             0 <= rcnum <= 32768
```

G.3. utcreradi — *Convert Reflectance/Temperature Image to Radiance Image*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*Create Radiance Image*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function converts a ground reflectance and, optionally, temperature distribution into a radiance image, including both reflected and radiated components. The reflected component uses only the reflectance image input (assumed Lambertian) with direct and diffuse illumination computed via MODTRAN in reference to the ASCII path data file. This file can be generated using the "MODTRAN Input" tool. It is important that the wavelength limits specified in the MODTRAN path data file cover each spectral band specified in the input image header. The direct, diffuse, or both illumination values from MODTRAN can be overridden. The radiated component is computed using both the reflectance and temperature (Kelvin units) images, and is ignored when the latter does not exist. The output image has radiance units (W/m3sr).

Command Line Arguments

```
Usage for utcreradi: Convert Reflectance/Temperature Image to Radiance Image
% utcreradi
  -i_refl     (infile)  Name of input IBSM reflectance image file
  -o_radi     (outfile) Name of output IBSM radiance image file

  [-i_temp]   (infile)  Name of input IBSM temperature image file (optional)
              (default = {none})
  [-workdir]  (string)  Optional working directory for MODTRAN
              (default = {none})
  [-i_mod]    (infile)  Name of input ASCII MODTRAN path data file
              (default = {none})
  [-direct]   (float)   Direct source irradiance at target in uW/cm2um (override)
              (direct >= 0.0, default = 0)
  [-zenith]   (float)   Source zenith angle in degrees (used with the direct field)
              (unbounded, default = 0)
  [-diffuse]  (float)   Diffuse downwelling radiance in uW/cm2umsr (override)
              (diffuse >= 0.0, default = 0)
```

G.4. utgrdangl — *Transform of image between ground and line-of-sight angle domains*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*Ground/Angle Transform*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function performs a coordinate transformation of a radiance image between the horizontal ground plane (meter units) and line-of-sight angular (radian units) coordinates. Only the image scaling attribute values are altered based on the viewing geometry. The defined azimuth axis is the non- squinted axis (normal to the line-of-sight and vertical axes). The scaling of the squinted axis is based on a linear projection, which occurs under the assumption that the ground field-of-view in the squinted direction is much smaller than the slant range. No image distortion or perspective effects are modeled.

Command Line Arguments

```
Usage for utgrdangl: Transform of image between ground and line-of-sight angle domains
% utgrdangl
  -i          (infile)  Name of input IBSM radiance image file
  -o          (outfile) Name of output IBSM radiance image file
  -range      (float)   Target slant range in km
               range >= 0.0
  -zenith     (float)   Target zenith angle in degrees (downlooking = 180)
               unbounded
  -azimuth    (cycle)   Definition of azimuth (non-squinted) axis
        0  "X",
        1  "Y"
```

```
-direction   (cycle)   Transform direction
        0  "Ground to Angle",
        1  "Angle to Ground"
```

G.5. utloadvars — *Load Cantata Variables from a File*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*Load Variables*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function loads a set of global variable definitions from a user specified ASCII file. The file format parallels the global variables input form, and can include defining variables as constants, expressions, and comments (starting with #).

Command Line Arguments

```
Usage for utloadvars: Load Cantata Variables from a File
% utloadvars
  -i          (infile) Loads the Cantata variables from the input file
```

G.6. utxform — *Transforms IBSM Files to/from Image/Spectrum*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*IBSM Transform*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function performs conversions between the image and spatial spectrum domains via 2-D Fast Fourier Transform (FFT) techniques. In addition to the FFT, the function performs scale factor conversion of the header data, quadrant swapping in both domains, and zero padding to the transform size. The transform size must be a power of two in both directions, and must be greater than or equal to the corresponding input image or spatial spectrum size.

Command Line Arguments

```
Usage for utxform: Transforms IBSM Files to/from Image/Spectrum
```

```
% utxform
  -i          (infile)  Name of input IBSM image or spatial spectrum file
  -o          (outfile) Name of output IBSM spatial spectrum or image file
  -nnx        (integer) Transform size (samples) in X direction
               1 <= nnx <= 16384
  -nny        (integer) Transform size (samples) in Y direction
               1 <= nny <= 16384
  -otype      (cycle)   Output data type
          1  "Complex",
          2  "Real",
          3  "Imaginary",
          4  "Magnitude",
          5  "Phase"

  -xform_dir   (cycle)   Transform direction
          1  "Spectrum to image",
          2  "Image to spectrum"
```

G.7. utxgraph — *Generate a command line to drive Xgraph*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*Xgraph Plot*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function provides a user interface to the Xgraph plotting application and allows up to four plots per graph with preformatted labels and axis ranges. The data is input as separate ASCII files in two column format.

Command Line Arguments

```
Usage for utxgraph: Generate a command line to drive Xgraph
% utxgraph
  [-i1]       (infile) First Xgraph input file
              (default = {none})
  [-lbl1]     (string) Input file #1 label
              (default = {none})
  [-i2]       (infile) Second Xgraph input file
              (default = {none})
  [-lbl2]     (string) Input file #2 label
              (default = {none})
  [-i3]       (infile) Third Xgraph input file
              (default = {none})
  [-lbl3]     (string) Input file #3 label
              (default = {none})
  [-i4]       (infile) Fourth Xgraph input file
              (default = {none})
  [-lbl4]     (string) Input file #4 label
```

```
            (default = {none})
[-i5]       (infile) Fifth Xgraph input file
            (default = {none})
[-lbl5]     (string) Input file #5 label
            (default = {none})
[-i6]       (infile) Sixth Xgraph input file
            (default = {none})
[-lbl6]     (string) Input file #6 label
            (default = {none})
[-i7]       (infile) Seventh Xgraph input file
            (default = {none})
[-lbl7]     (string) Input file #7 label
            (default = {none})
[-i8]       (infile) Eighth Xgraph input file
            (default = {none})
[-lbl8]     (string) Input file #8 label
            (default = {none})
[-logx]     (flag)   Flag to select logarithmic X axis
[-logy]     (flag)   Flag to select logarithmic Y axis
[-xlabel]   (string) Text for X axis label
            (default = {none})
[-ylabel]   (string) Text for Y axis label
            (default = {none})
[-title]    (string) Text for graph title
            (default = {none})

[Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
  [-xmin] (float) Minimum X scale value
          (unbounded, default = 0)
  [-xmax] (float) Maximum X scale value
          (unbounded, default = 1)

[Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
  [-ymin] (float) Minimum X axis scale value
          (unbounded, default = 0)
  [-ymax] (float) Minimum X axis scale value
          (unbounded, default = 1)
```

G.8. viff2ibsm — *Converts a VIFF File to IBSM Format*

Object Information

Category:	*IBSM*	Subcategory:	*Utilities*
Operator:	*VIFF to IBSM*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

This function adds attribute information to a VIFF file to create an IBSM format output. The spatial scaling and spectral band data can either be entered from the screen (up to four bands), or input from an ASCII header file. The file format corresponds to that generated using the "List IBSM File Data" function.

Command Line Arguments

```
Usage for viff2ibsm: Converts a VIFF File to IBSM Format
% viff2ibsm
  -i       (infile)  Name of input VIFF image
  -o       (outfile) Name of output IBSM image
  -xmin    (float)   Minimum X value (normally rad or rad-1 units)
           unbounded
  -xmax    (float)   Maximum X value (normally rad or rad-1 units)
           unbounded
  -ymin    (float)   Minimum Y value (normally rad or rad-1 units)
           unbounded
  -ymax    (float)   Maximum Y value (normally rad or rad-1 units)
           unbounded
  -wl1     (float)   Center wavelength of band #1 in microns
           unbounded
  -bw1     (float)   Spectral bandwidth of band #1 in microns
           bw1 > 0.0
  -wl2     (float)   Center wavelength of band #2 in microns
           unbounded
  -bw2     (float)   Spectral bandwidth of band #2 in microns
           bw2 > 0.0
  -wl3     (float)   Center wavelength of band #3 in microns
           unbounded
  -bw3     (float)   Spectral bandwidth of band #3 in microns
           bw3 > 0.0
  -wl4     (float)   Center wavelength of band #4 in microns
           unbounded
  -bw4     (float)   Spectral bandwidth of band #4 in microns
           bw4 > 0.0

  [-iasc] (infile)  Name of input ASCII header file (optional)
          (default = {none})
```